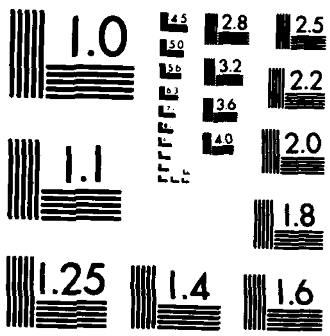
AD-A151 813 THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 1/3

UNCLASSIFIED T R VERMILLION DEC 84 AFIT/GE/ENG/84D-68 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTER-AIDED DESIGN TOOLS INTO A TOOLKIT OPTIMIZED FOR ACADEMIC APPLICATIONS

THESIS

AFIT/GE/ENG/84D-68 Thomas R. Vermillion
Captain USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/84D-68

THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTER-AIDED DESIGN TOOLS INTO A TOOLKIT OPTIMIZED FOR ACADEMIC APPLICATIONS

THESIS

AFIT/GE/ENG/84D-68 Thomas R. Vermillion
Captain USAF

Approved for public release; distribution unlimited.

AFIT/GE/ENG/84D-68

THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTER-AIDED DESIGN TOOLS INTO A TOOLKIT OPTIMIZED FOR ACADEMIC APPLICATIONS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Thomas R. Vermillion, B.S.
Captain USAF
Graduate Electrical Engineering

December 1984

Approved for public release; distribution unlimited.

Preface

I began my search for a thesis topic with the intention of identifying and solving an engineering problem of international scope. Fortunately, those more learned and experienced in the solution of engineering problems directed me toward a more realistic goal. The desire to address a substantive problem was encouraged and remained.

The need to develop a completely integrated set of VLSI CAD tools became clear to me in the wee hours of a Spring morning as I struggled to complete a VLSI design project using two incompatible CAD tools.

Perhaps this work is not as revolutionary as I had envisioned but I believe it provides a solid basis for the development of the elusive integrated toolkit of VLSI CAD tools. I have learned much and have developed a high regard for those who have made VLSI technology available to all of us.

Many of the ideas developed during the characterization of the CAD toolkit were developed during a review of Paul Losleben's excellent and comprehensive paper on VLSI [17].

I am indebted to Hal Carter, my thesis advisor, for permitting this project and for his enthusiastic approach to my work, his concise suggestions, and tactful criticisms. I offer a special thanks to Bill Sutton for his prompt and careful reviews of the segmented drafts that preceded this final product.

Tom Vermillion

Contents

List of Figures

AFIT/GE/ENG/84D-68

Abstract

The literature indicates that the success of future VLSI design efforts is contingent upon the evolution of a new design philosophy and upon the production of a special kind of engineer, a VLSI design engineer. The lack of an integrated, compatible set of VLSI CAD tools has been suggested as a serious problem which prohibits the optimization of the academic curricula necessary to produce the required VLSI design engineers.

This thesis proposes that the CAD tool problem may be solved through the development of a VLSI CAD "toolkit". The characteristics of an integrated, technically complete, academically-oriented VLSI CAD toolkit have been presented. A systematic method to integrate CAD tools was developed and supported through the design of a set of CAD tool evaluation metrics. Finally, a prototype VLSI CAD toolkit and its custodian were designed, implemented, and evaluated.

The characterization of the integrated toolkit, though not complete, appears to constitute a solid basis for future work. The evaluation metrics and methodology were simple to implement and provided absolute and relative measurements of the "level of presence" of desired characteristics in toolkit components. The evaluation metrics show promise but further application of the metrics to collections of CAD tools will be required if a high level of confidence is to be assured. Suggestions supplied by [illegible] → p. 147 ([illegible])

THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTER-AIDED DESIGN TOOLS INTO A TOOLKIT OPTIMIZED FOR ACADEMIC APPLICATIONS

I. Introduction

Background

The successful defense of the United States is predicated upon a deterence of conflict through possession of military systems that are more capable and survivable than those possessed by potential opponents. Today, the enhancement of military systems primarily requires the reduction of their cost, size, weight, and power requirements while simultaneously increasing their speed, intelligence, reliability, and survivability. The enhancements are being implemented chiefly through the replacement of manual and semiautomatic electro-mechanical sub-systems with intelligent, fully automatic, electronic modules.

Current technology provides the capability to construct electronic modules composed of many active and passive electronic devices arranged on a very small piece or chip of silicon. The silicon electronic modules require very little power and are housed in extremely small packages. These microelectronic modules, implemented through the integration of many electronic devices, are known as integrated circuits (ICs). Integrated circuits are classified according to the

density of individual devices integrated on a single silicon chip. The generally agreed upon classifications of IC integration are:

Small Scale Integration	(SSI)	: 1 - 10 dev.
Medium Scale Integration	(MSI)	: 10 - 100 dev.
Large Scale Integration	(LSI)	: 100 - 100,000 dev.
Very Large Scale Integration	(VLSI)	: 100,000+ dev.

One author proposes an expansion of the IC classification system to accommodate a new classification, Ultra Large Scale Integration (ULSI) [27]. The expanded classification system defines VLSI as 200,000+ devices and ULSI as 10,000,000+ devices.

The development and integration of VLSI circuits into military systems are essential elements of the successful defense of the United States. The U.S. Department of Defense (DOD) has underscored this assertion through its $400 million program for the research and development of Very High-Speed Integrated Circuits (VHSIC). The VHSIC program has been called the most ambitious and important Federal program since the United States began the exploration of space [33]. The VHSIC program's goal is the development of high-speed, reliable, silicon chips that will ensure continued U.S. superiority in defense electronics [33].

The implementation of VLSI-based defense systems will necessitate many changes within the DOD and within the organizations and companies that provide support to the DOD. Perhaps the most important changes will be required in academic institutions where curricula must be revised to

produce a new type of engineer, the VLSI design engineer [6].

Paul Drongowski, in a paper describing the VLSI curriculum at Case Western Reserve University, suggests that VLSI design engineers fall into two categories: 1) "tall-thin" engineers capable of spanning VLSI design from architecture to circuit analysis, and 2) "short-fat" engineers who are particularly adept at some specific design task [9]. The successful design of future VLSI circuitry requires the education and hiring of a new class of engineers, "tall-thin" VLSI engineers, with broad backgrounds in electronics, system architecture, software, and computers.

The early development of integrated circuitry was accomplished by people primarily skilled in the physics and materials of solid state devices. The initial growth of IC density (hence functional complexity) was much slower than that being experienced today. Historically, the design of ICs and the translation of IC designs into actual hardware has been done by teams of "short-fat" engineers using manual techniques. The teaching and use of manual design and implementation techniques has been a satisfactory, although laborious, method of creating SSI, MSI, and LSI integrated circuits. Unfortunately, previous IC design methods and implementation techniques are less effective with VLSI designs.

The increased functional complexity and device density (with its attendant reduction in geometrical tolerances)

associated with VLSI circuitry have necessitated a reevaluation of the current approaches to IC design and production. It is generally recognized that computer-aided design (CAD) equipment and programs are critically important elements of future VLSI design methodologies [4,17,19,28,34,40]. Educational institutions are presently faced with rapidly implementing new curricula to address these design issues at a time when the industrial community is attracting the most qualified teachers [33]. The U.S. Air Force, and its engineering school, the Air Force Institute of Technology (AFIT), have not escaped the need for curricula changes nor the loss of qualified instructors.

AFIT has expanded its graduate-level electrical-engineering curriculum to include courses and individual research efforts designed to produce engineers and managers qualified in the design and evaluation of VLSI circuits and systems. The importance of automated design and evaluation philosophies and their associated techniques are fundamental themes of each VLSI course, laboratory, and research project. The success of the AFIT VLSI curriculum rests in the ability to expose engineers and managers to the methodologies and processes associated with VLSI design in a very short period of time (typically 3 courses). The availability of high quality VLSI CAD programs is essential if AFIT students are to rapidly and effectively acquire a useful background in VLSI design.

The current generation of CAD programs, often referred

to as "tools", evolved during independent academic, commercial, and DOD projects. Those concerned with VLSI circuitry developed and implemented the CAD tools necessary to achieve their specific goals. This independent growth of CAD tools has led to the establishment of generally disjoint sets of application-specific tools with various levels of compatibility. The willingness of several universities to share their locally developed tool collections with AFIT has substantially decreased the time needed to develop a VLSI curriculum at AFIT. Although the use of these tools has been very beneficial over the past years; a problem has been observed which must be corrected if the AFIT VLSI curriculum is to fully achieve its goals.

Problem

The problem is twofold. First, the tools, which are widely varied in form,function, and complexity, require too much time to learn their effective use. Second, design projects, the most highly productive and instructive components of the VLSI curriculum, are limited to a low level of complexity. The projects are limited because the systematic access to the broad spectrum of CAD tools, essential for realistic, complex VLSI design efforts, is not possible with the the current arrangement of AFIT CAD tools. Although there is a wide range of powerful CAD tools available to AFIT students, the tools reside in different, and frequently incompatible, tool collections. The time required to com-

pensate for tool incompatibilities inhibits student use of many tools.

The basis of the problem is the lack of an integrated, compatible set of VLSI CAD tools, which may be easily learned, and utilized. This serious problem prohibits the optimal use of the time available to produce the high quality VLSI engineers and managers necessary to meet the challenges facing the DOD in the next few years.

The solution to the problem is the integration of available VLSI CAD tools into a technically complete "toolkit" which is easy to learn, to use, to update, and to manage.

This thesis will describe the characteristics and components of an integrated, technically complete, academically-oriented VLSI CAD toolkit. The attributes of an academically sound VLSI CAD toolkit and its components will be identified. A set of evaluation metrics (mathematical relationships to measure and weight the desirable attributes) will be developed. Furthermore, a systematic method, based upon the attributes and metrics described above, will be proposed to enable selection of the most suitable toolkit components from a generally disjoint VLSI CAD tool inventory.

Assumptions

Several assumptions have been made to reduce the scope of this thesis to a level manageable in the few months available. It is assumed that:

1. That the reader has an understanding of the fabrication processes used to create the fundamental components of silicon-based integrated circuits.

2. That the reader has a good background in semiconductor devices and technologies.

3. That the reader has a good background and working skills in at least one high-level computer language that permits the use of structured programming techniques.

4. A subjectively developed set of CAD tool attributes and evaluation metrics will be acceptable to the reader as rigorous and comprehensive enough to demonstrate that the toolkit component selection methodology is sound.

5. Tailoring the tool evaluation metrics to suit the needs of academic institutions does not nullify their value when using them to evaluate commercially-developed CAD tools.

6. The characteristics of a VLSI CAD tool can be adequately observed by using a simple test input and sequentially exercising all of the tool options on that input.

7. Discrepancies and deficiencies noted in an evaluated tool are caused by the tool structure as advertised and documented and not by improper installation of the tool or undetected local modification of the tool.

<u>Summary of Current Knowledge</u>

The critical importance of CAD tools to the successful design of VLSI circuitry is well documented. CAD software

has been referred to as the "backbone" of modern VLSI design [28]. One theme present in much of the literature reviewed is that continued CAD tool improvements are absolutely essential to make VLSI design available to people not necessarily skilled in semiconductor technologies. Access to VLSI design by the disciplines that support or use VLSI circuitry is required if the major problems impeding VLSI evolution are to be resolved. [13,20,31].

Much work has been reported concerning the development and improvement of specific CAD tools. Several discussions of the future of VLSI design address the need to develop a connectivity between the various design levels (usually in the form of a centralized database).

The proceedings of recent design automation (DA) conferences indicate there are three major areas where work is being concentrated: 1) identification and refinement of technologies that permit an increase in IC component density, 2) development of specific CAD tools to cope with increased design complexities and 3) identification of IC design methodologies which will increase the productivity of IC designers.

Many efforts to establish metrics for software evaluation have been reported. Most of the metrics observed were based upon the detailed constructs inherent in a particular language or class of languages. One particularly valuable source of metric development information was found in a report prepared by Dean and McCune concerning the evaluation

of advanced tools for software maintenance [8].

Few efforts were found that addressed the specific, central issue of this thesis, a methodology for VLSI CAD tool integration. One effort to integrate engineering design tools, a project to create a Designer's Workbench by Bell Laboratories, was reviewed. Several useful ideas have been found in a report on the delivery of CAD tools for the Designer's Workbench [13]. If a single philosophy or viewpoint seems to prevail throughout this thesis, it has emerged from a review of the excellent, comprehensive discussion of VLSI CAD tools prepared by Paul Losleben [17].

Approach

This thesis approaches the IC CAD tool integration from an academic viewpoint. The methods developed are intended for application in the academic arena. However, some may be useful to the industrial community.

First, past and current discussions concerning VLSI design methods will be reviewed. Next, a review of published literature and documentation pertinent to the use and evolution of VLSI-oriented CAD tools will be performed. The information obtained from the reviews will provide the basis for the development and presentation of: 1) Support for a VLSI design philosophy which incorporate an integrated CAD toolkit, 2) the metrics necessary to evaluate a VLSI CAD toolkit and its component tools, and 3) a methodology to use the metrics developed to construct an integrated, VLSI CAD

toolkit which will adequately support aggressive VLSI curricula.

The metrics and methodology developed will be evaluated through a partial application of them to the VLSI CAD tools currently available at AFIT. A prototype toolkit will be suggested to support the soundness of the methodology. Missing tools will be identified and the high level design of the program necessary to control the toolkit will be completed.

Finally, recommendations will be made for future work concerning the integration of VLSI CAD tools.

Sequence of Presentation

Chapter II opens with a brief chronology of the evolution of VLSI circuits and a review of IC design methodologies. The prevailing IC design philosophy and its applicability to VLSI requirements is discussed. The essential characteristics of a viable VLSI design methodology are summarized. Finally, the critical VLSI design role of integrated IC CAD tools is established.

Chapter III establishes the general structure of an ideal kit of VLSI CAD tools and illustrates the use of the toolkit by outlining a "typical" design session.

Chapter IV completes the characterization of a useful, complete, maintainable, integrated VLSI CAD toolkit by presenting a detailed description of the ideal toolkit structure.

Chapter V details the development of the metrics that

will be used for toolkit and tool evaluation. The mechanics of CAD tool integration are also presented.

Chapter VI provides a discussion of the work done to create the examples provided in the appendices and should serve as a first-level evaluation of the validity of the integration method presented herein.

Chapter VII Provides summary of this author's work and recommendations for future work concerning the integration of IC CAD tools.

II. VLSI Design Methods

Chronology of VLSI Development

The evolution of the integrated circuit, and certainly the VLSI circuit, was not a miraculous occurrence but was a rather difficult effort marked by many engineering and scientific milestones.

Several significant milestones in the evolution of VLSI circuits are summarized below [35].

1947 (Dec): Bell Laboratory physicists Walter H. Brattain and John Bardeen, following suggestions made by physicist William Shockley, demonstrated electrical amplification by using the properties of electric fields near the surface of a germanium-based "point-contact" device. John R. Pierce calls the device a "transistor".

1947: Immediately following the demonstration of the "point-contact" transistor, Shockley suggests the possibility of transistor action in a structure that sandwiched an n-type semi-conductor between two p-type semiconductors. Shockley calls the structure a "junction transistor". Shockley could not prove his theory because there was no way to construct the "junction transistor" at that time.

1948 (June): The military decides there is no need to classify the transistor and Bell Laboratory holds first public demonstration of the transistor. Public response is somewhat indifferent.

1953 (June): Through the leadership of the Army Signal Corps, standards for transistor operating characteristics were proposed.

1954: Texas Instruments Inc. (TI) announces the first transistor made of silicon.

1956: Shockley, Brattain, and Bardeen are awarded the Nobel Prize in physics for research in semiconductors and for the invention of the transistor.

1957: John T. Wallmark, working at Radio Corporation of America (RCA) laboratories, is awarded a patent for the field-effect transistor (FET).

1958: Fairchild Camera and Instrument Corporation produces the first device built by diffusion, the diffused-base, mesa transistor. The device was called a mesa transistor because of the "stacked" construction of the transistor's base, emitter, and collector regions.

1958: Jean Hoerni, a physicist with Fairchild, invents the planar process for transistor construction and suggests a metal deposition technique for component inter-connections.

1958: Jack Kilby, an engineer working for TI, assembles the first "integrated circuit" (IC). The all silicon circuit was a phase-shift oscillator constructed with mesa transistors.

1959 (January): Robert Noyce of Fairchild Semiconductor Inc. enters in his notes a description of an "integrated circuit" which would incorporate diffused resistors and evaporated metallic interconnections. Noyce and Kilby are considered independent inventors of the IC.

1959 (March): Kilby designs and builds a flip-flop using monolithic germanium. The "solid circuit" was unveiled at the Institute of Radio Engineers show.

1961: The first commercially available monolithic integrated circuit is marketed by Fairchild. The IC was a flip-flop built using resistor-transistor logic (RTL) and contained four bipolar transistors and two resistors.

1962: Steven Hofstein and Frederic P. Heiman, young engineers working for RCA, produced the first metal oxide semi-conductor FET (MOSFET). They demonstrate the usefulness of the MOSFET by building a MOSFET-based, multi-purpose logic block. The logic block contained 16 MOSFETs on a 2,500-square-mil-chip of silicon.

1965 (August): Fairchild announces the dual in-line (DIP) IC package.

1965: Robert Widlar, a designer with Fairchild, develops the first practical integrated circuit operational amplifier (opamp), the uA709. Widlar also designed the uA702, uA710, and the uA741

opamps.

1966: Autonetics delivers the first silicon-on-sapphire (SOS) chip to the Massachusetts Institute of Technology. The chip was a matrix of 6,144 diodes for a character generator in a display.

1967: Fairchild announces the first read-only memory (ROM) chip. The ROM was a 64 bit MOS memory.

1971: Intel Corporation unveils the first micro-processor(uP), the 4 bit 4004. The chip was a silicon-based, p-channel MOS design and measured 110 by 150 mils.

1971: Intel Corporation announces the first 8-bit microprocessor, the 8008. The chip used MOSFET circuitry and contained approximately 2,900 devices.

1979: IBM announces the first 64-K (65,536 bits) dynamic random-access memory. This chip has been called the first VLSI circuit.

Approaches to the Design of Integrated Circuitry

The evolution of integrated circuitry has been accompanied by a non-linear increase in chip complexity. In 1964, Gordon E. Moore, then director of research at Fairchild Semiconductor Inc., was the first to predict the growth of IC complexity when he suggested that the complexity would double every year [26]. No significant departure from that prediction has been observed. In fact, Moore's prediction is often referred to as "Moore's Law" [17,26]. Demands upon the resources required to accomplish an IC design have increased in a non-linear manner proportional to the increase in IC complexity. The design philosophies and approaches used to manage and develop IC design resources have not matured sufficiently to meet VLSI requirements. The

remainder of this chapter focuses on the current approaches to IC design and what must be done to extend these approaches to encompass VLSI design requirements.

<u>The Early Design Approaches</u>. The approach taken to design the first IC in 1958 prevailed through the mid 1970s. The experience necessary to accomplish electronic design resided primarily in those scientists and engineers familiar with solid-state physics [20]. Electronic design was an art applied to the sciences that provided the basis for electronics [18]. The early IC chips were designed by one or two people who had few constraints placed on their designs or their design methodology [17]. Communications between IC designers and the designers of systems employing ICs were very poor consequently, both groups established essentially independent goals. Most major IC design efforts were projects to assemble a family of chips that encompassed all the basic functions of boolean logic. Few chips contained complex functions and system designers were faced with the task of integrating large numbers of the SSI ICs to perform a complex task. The IC development process from conception to production was a manual one.

The design process is simply summarized. A chip's architecture was dictated by the desired logic function. Once a function had been selected, the function variables were manually minimized and translated into logical and electrical diagrams. The resulting electrical circuit was physically built on a "brass-board" and tested. If the

tests proved successful, the layers representing the electrical elements were manually drawn or "layed-out" on a grid that represented the surface of the silicon chip. Masks were created by copying the layer artwork onto a material known as Rubylith (a polyester-base covered by a strippable red plastic). The red plastic was manually removed in those areas where exposure of the silicon was desired. The Rubylith artwork was photo-optically reduced to produce masks suitable in size for direct contact masking of the silicon chip. The silicon chip was exposed via a sequential application of the masks and a planar-processed IC resulted.

Manual drafting and Rubylith artwork were the primary design tools well into the 1970s [17]. In the mid-1970s the complexity of ICs reached a level that necessitated drastic changes in the approaches to IC design.

There began a movement from the "single" designer approach to a design team approach. The design teams were composed of "short-fat" engineers, engineers highly specialized in one facet of IC design. A stratified approach to design was initiated and the teams were assembled in compatible hierarchies to address the distinct levels associated with the design. Large companies automated some of the design tasks (notably logic minimization and mask generation) and design automation (DA) became a major discipline in computer science. The changes were not without consequence. Creation of the design teams were accompanied by the conflicts inherent with the division of labor [24].

There was a lack of communication between adjacent design hierarchies which led to wasteful iterations of design steps. The separation between chip designer and system designer continued. There was resistance to the implementation of automated design aids [17,40]. The reluctance to use CAD programs was partly due to numerous software failures [17]. The advancement of DA was hindered by proprietary interests of companies developing in-house design aids [18].

In spite of the problems, IC design progressed during the 70s and the chip complexities grew in accordance with "Moore's Law". Sophisticated chips, microprocessors and memories, emerged which found applications in nearly every industry. With the appearance of the "first" VLSI chip in 1979 came the realization that major improvements in IC design management and design automation techniques were required. That realization was the seed of the dynamic change that characterizes the current approaches to IC design.

Current Design Approaches. Today's methods of IC design are widely varied, rapidly changing, refinements of the approaches that emerged from the major modifications of the mid 1970s. Present industrial and academic approaches to IC design incorporate semi-automated improvements of early IC design methods.

Industrial IC design is evolving in a manner aimed at

minimizing what has been called the biggest deterent to VLSI development today, non-recurring design costs [6]. The primary goal of minimizing non-recurring design costs is but one of the four chief goals of a modern design team. The remaining industrial design team goals are: 1) to shorten the product development cycle, 2) to minimize design risks, and 3) to produce an error-free design [6].

The academic approach to IC design does not radically differ from that found in industry but the differences are worthy of discussion. The objective of modern academic IC design curricula is to produce qualified students capable of: 1) performing VLSI-oriented research, and 2) succeeding as industrial VLSI designers [9]. Preparation of students for VLSI research environments necessitates the use of a VLSI design approach which emphasizes the flexibility (or inflexibility) of the design approach itself. These students will be the people who produce the innovative changes necessary to continue effective evolution of the VLSI design process. Design approaches used in research-oriented curricula may vary, perhaps from one course to another. Research-oriented curricula may suggest approaches to VLSI design that are "risky" in the industrial sense but which may lead to the creation of innovative changes to the design process. The preparation of students for a successful industrial career requires curricula with VLSI design philosophies which parallel those found in industrial VLSI design. Most curricula contain a combination of both approaches.

Although current industrial and academic VLSI design approaches differ in the importance placed upon specific design objectives, their design approaches have many commonalities. Decomposition of the design process into its fundamental levels of abstraction is the key to understanding the current design philosophies.

The general design process may be decomposed from many different viewpoints. One decomposition details two fundamental levels or structures in the design process: functional design and physical design [17]. Another decomposition identifies five domains or levels of abstraction: 1) the architectural level, 2) the organizational level, 3) the logic level, 4) the geometry level, and 5) the electrical level [9]. There are other decompositions [40]. The commonalities of modern industrial and academic VLSI design approaches, hence the current design philosophy, have been illustrated below through the application of the various decomposition schemes to the design approaches.

A modern VLSI design task is usually approached at two levels: a functional design level and a physical design level. The design team approach is used both by industry and academic institutions. It is not unusual for industry to have design teams at each level or several design teams within these levels. The academic approach, which encourages the development of "tall-thin" designers, seems to favor a single design team spanning both levels. At each level, the design process is hierarchical in nature. That

is, design problems are successively decomposed into smaller problems while synthesizing small design elements into larger functional elements [9]. The decomposition and synthesis are iterated until an acceptable design is achieved.

The functional design level includes the architectural and logical design tasks. Current approaches to these tasks are essentially manually implemented. The architectural task is concerned with: 1) the specification of system behavior, 2) the identification of data and control flow throughout the system, and 3) identification of system-level test requirements. The goal at the architectural level is to provide a configuration that best satisfies the specified behavior [40]. The logical design task is also three-fold: 1) the desired system behavior is translated into a set of primitive building blocks sufficient to implement the desired behavior, 2) the primitive building blocks are synthesized into the largest blocks possible, and 3) the logical design is tested to verify implementation of the desired behavior. The logical design effort should produce logic diagrams and/or boolean equations which implement the desired behavior and whose primitive implementations are available using current VLSI technology [40]. In summary, the functional design level should produce a satisfactory specification of desired system behavior and the logical means to implement the behavior using current VLSI technologies.

Most of the changes seen in the evolving IC design

process have occurred at the physical design level and were initiated to address IC complexities at the VLSI level. The physical design level includes the following design tasks: 1) floor plan development, 2) layout or translation of the logical design blocks into physically equivalent device assemblies, 3) interconnection of the on-chip devices, 4) topological analysis of the chip layout, 5) timing and electrical analysis of the chip layout, 6) verification that the layout implements the specified logical design, and 7) translation of the chip layout information into a form usable by chip fabrication facilities. Many of the physical design tasks have been automated or are supported by CAD tools. The physical design level should produce the data necessary to correctly fabricate an IC that implements the original system specification. Many contributions have been made by the academic community at the physical design level. Most of these contributions have been in the development of CAD tools to simplify the various design tasks.

The design methods summarized above were best described by M.F. Oakes when he said: "Today's methods have not been designed, they have evolved" [27].

Future Design Approaches. The future of VLSI hardware rests in the number of electronic applications where VLSI offers a cost-effective alternative to existing electronics. The number of future applications for VLSI hardware will be functions of VLSI design costs (non-recurring costs in particular) and design time [17]. The needed reduction in VLSI

design costs and time can be achieved through the scientific DESIGN of a VLSI design methodology.

A complete characterization of the ideal VLSI design methodology has yet to be developed. Many published works propose design methods which address one or several of the issues which must be resolved if VLSI design is to meet future challenges. Many of the proposed methods share common components. A scientific evaluation of the common components should lead to a VLSI design methodology which will support the design complexities required by future IC applications. A review of the frequently proposed components of future VLSI design processes is provided below.

It is generally agreed that future VLSI design methods must be hierarchical in nature and that future methods must approach IC design from a system-level viewpoint [4,9,17,29,40]. Another essential characteristic of the ideal VLSI design process is a well-defined connectivity between the levels of design abstraction [4,9,18,27,29,31]. The usual approach proposed to provide the required connectivity is the use of a single, centralized database or data structure throughout the entire design process. A thorough and effective design management technique must be incorporated into the ideal design method. A recurring issue in proposed VLSI design techniques is the need to make VLSI design accessible to people whose skills lie in disciplines that either support or utilize VLSI circuitry. Finally, throughout the proposals for improvements in VLSI design

methods, one component surfaces as the most critical and essential element of the ideal VLSI design process: INTEGRATED CAD TOOLS [6,12,17,18,20,22,27,28,29,34,40,41]. It is believed that proper integration of IC CAD tools will open VLSI design to a broad range of disciplines that have major contributions to make to future VLSI development. An integrated set of CAD tools, sharing a common database, will provide the inter-level connectivity and design management capability essential for the success of future VLSI design efforts.

Academic institutions have long realized the importance of CAD tools to successful IC design. Consequently, many, if not most, of the IC CAD tools in use today were developed in the academic arena. The academic CAD tools are freely shared and improvements are nearly continuous. Carver Mead, a pioneer in the development of IC CAD tools, believes much of the thrust for new innovation in VLSI design is coming from the universities [22]. Thus, it seems appropriate that the universities should address and meet the CAD tool integration challenge.

<u>Summary</u> A chronology of VLSI evolution has been presented to develop an appreciation of the effort expended to reach production of the first VLSI chip. A summary of past and present IC design methods was provided to demonstrate the slow evolution of the IC design process from a single-designer, manual effort to a team-oriented, partially auto-

mated approach. The essential elements of future VLSI design methodologies have been examined and the need for integrated VLSI CAD tools is proposed as the most critical and essential of all the required elements. This thesis addresses the need for integrated VLSI CAD tools and will provide one systematic, perhaps scientific, method to integrate the tools.

III. General Characteristics of a VLSI CAD Toolkit

Introduction

The importance of VLSI circuitry to the DOD has been established. It has been shown that the design methodologies used to implement SSI, MSI, and LSI designs are not sufficient to meet VLSI design requirements. A review of current literature indicated that the emerging VLSI design methodology will emphasize automation of design tasks and that improved CAD tools are critically important to the success of future VLSI design tasks. Access to VLSI design for people with wide variations in technical backgrounds and increased designer productivity were stated as the primary goals of VLSI CAD tool development. Integration of VLSI CAD tools into a single toolkit was identified as a central task in the development of a VLSI design methodology that will meet academic requirements.

This chapter describes the general properties and characteristics of an integrated, academically-oriented, VLSI CAD toolkit and its components. Also, a typical design session using the toolkit is outlined. The toolkit described below and in subsequent chapters is not the ultimate VLSI CAD toolkit. The intention here is to develop a basis for systematic CAD tool integration with emphasis placed upon academic requirements.

General Properties of VLSI CAD Toolkits and Toolkit Components

Throughout the remainder of this section, references to

a VLSI CAD toolkit are also references to individual toolkit components.

A VLSI CAD toolkit must be useful, complete, and maintainable. These toolkit properties are examined below.

A Useful VLSI CAD Toolkit/Toolkit Component. A VLSI CAD toolkit may be considered useful if it possesses the following characteristics:

1. The toolkit's capabilities, limitations, input formats, output formats, and command syntax are easy to learn.

2. The toolkit is easy to select and execute.

3. The toolkit is easy to install on the host computer system.

4. The toolkit substantially increases the productivity of the user.

5. The toolkit effectively uses design resources.

6. The toolkit makes incompatibilities in subordinate components transparent to the user.

7. The toolkit provides a mechanism to control design information, correctness, and security.

8. The toolkit can be used effectively by people with wide variations in design abilities and technical expertise.

A Complete VLSI CAD Toolkit. The VLSI design methodology reviewed in chapter 2 was decomposed into two fundamental design levels: a functional level and a physical level. The functional and physical levels were further decomposed into several, sub-ordinate design levels. A complete VLSI CAD toolkit possesses the characteristics of a useful toolkit at each level of design.

A Maintainable VLSI CAD Toolkit. A maintainable VLSI

CAD toolkit is one which includes mechanisms to prevent degradations in the toolkit usefulness and completeness. In particular, a VLSI CAD toolkit is maintainable if it possesses the following characteristics:

1. Additional tools/features may be easily added.

2. Component tools/features may be easily modified without changing the fundamental toolkit structure.

3. Toolkit documentation contains the following information:

    a. Documented source code.

    b. Diagrams that depict the overall structure of the toolkit.

    c. Diagrams that depict the flow of information through the toolkit.

    d. Diagrams that depict the flow of control through the toolkit.

    e. Implementation specific information concerning the toolkit.

5. Toolkit documentation may be easily updated, reproduced, and distributed.

6. The toolkit is transportable between similar computer systems.

7. Test cases and examples sufficient to demonstrate both the operation of the toolkit and the tasks necessary to perform all maintenance operations.

General Organization of a VLSI CAD Toolkit

The VLSI CAD toolkit organization presented below represents a toolkit configuration which is useful, complete, and maintainable.

One way to describe the organization of a VLSI CAD

toolkit is to develop an analogy between a VLSI CAD toolkit and toolkits commonly found in modern workshops.

A toolkit may be represented as a collection of drawers, each of which contains tools useful for a particular task or a group of closely related tasks. Access to the drawers and their contents is controlled by a toolkit custodian.

A VLSI CAD toolkit may be viewed as a collection of VLSI CAD tools which reside in "drawers" and which are controlled by a single custodian. The toolkit drawers may be partitioned into three groups: 1) design management tools, 2) design tools, and 3) information storage. Figures 1 and 2 illustrate such a toolkit and depict the custodian's span of control, respectively. The contents of each drawer partition and the general responsibilities of the toolkit custodian are described below.

<u>The Toolkit Custodian</u>. The custodian of a VLSI CAD toolkit must:

1. Provide the only user interface to the toolkit.

2. Control access to each toolkit drawer.

3. Control access to each drawer component.

4. Control the flow of information into, within, and out of the toolkit.

5. Control the sequence in which toolkit drawers are opened and closed.

6. Control the sequence in which drawer components are used.

7. Control the parameters used during the execution of component tools.

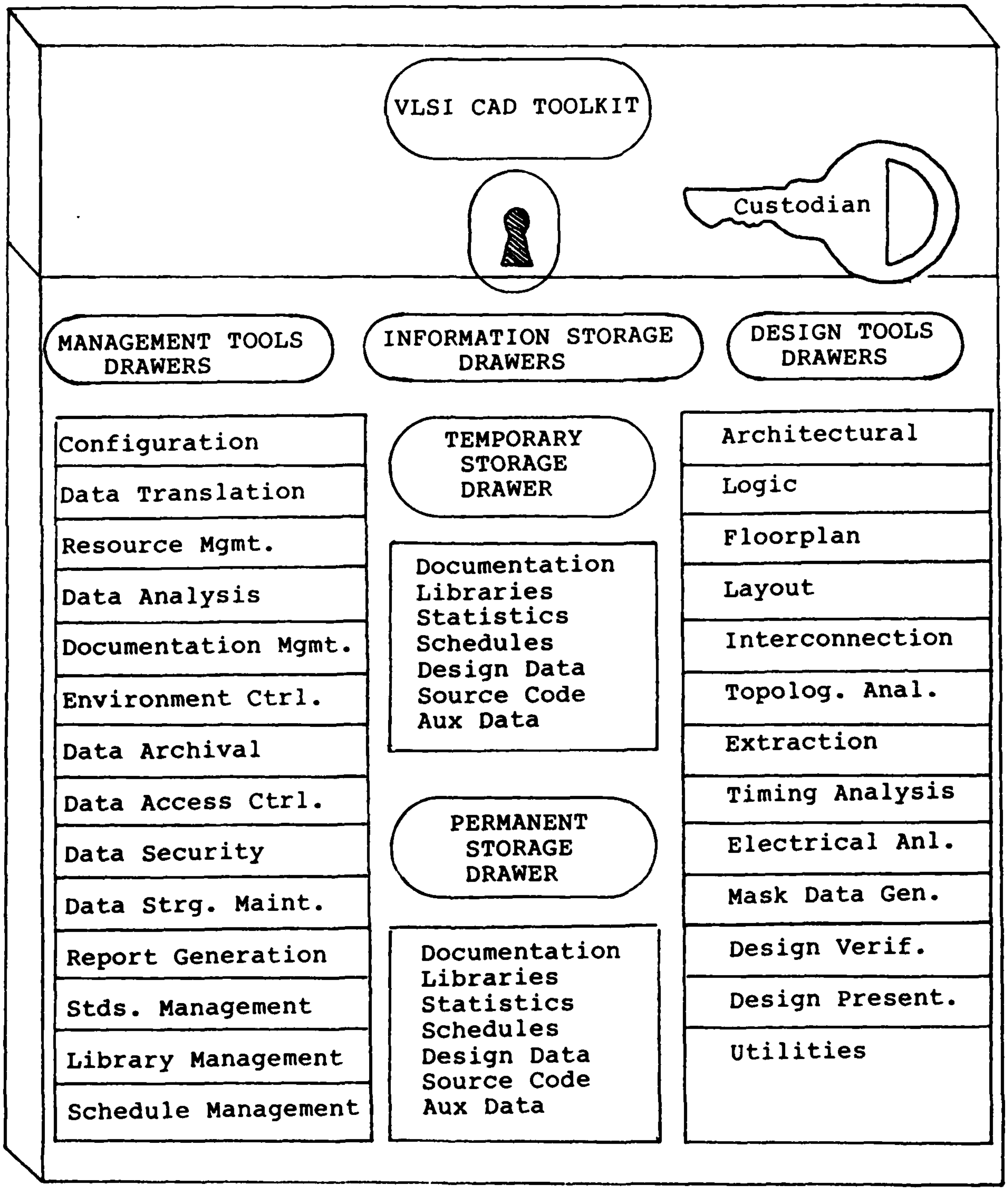


Figure III-1 VLSI CAD Toolkit Characterization

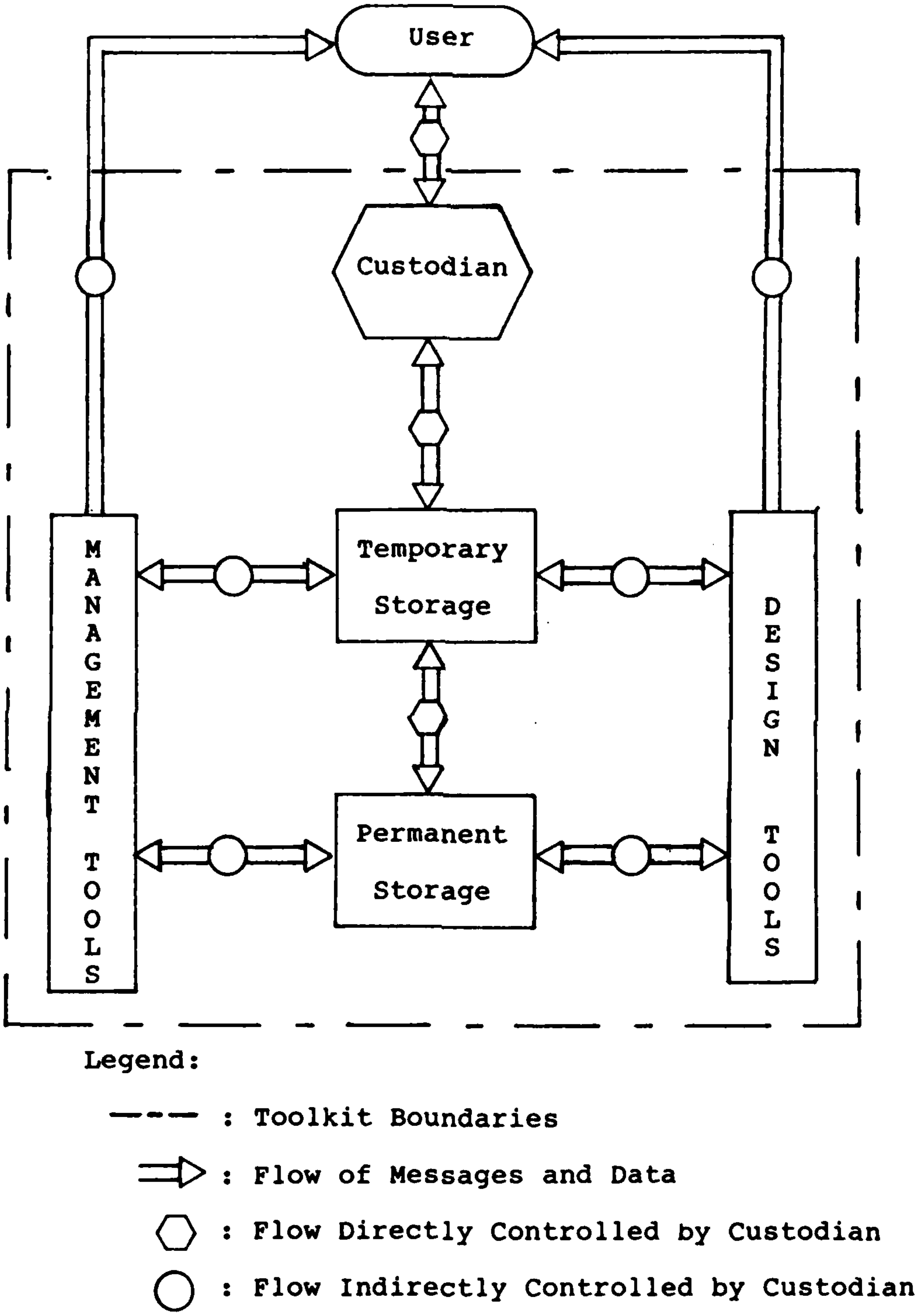


Figure III-2 Custodian Span of Control

8. Control additions and modifications to the toolkit drawers and their components.

Design Management Drawers. The toolkit must include drawers containing the tools necessary to accomplish the design management tasks listed below. [2,17]

1. Configuration Management: Management of the toolkit structure, operating system interface, and component relationships.

2. Data Translation: Translation of input/output data into useful formats.

3. Resource Management: Management of the toolkit's use of computer system resources.

4. Data Analysis: Analysis of data to ensure compatibility with user and toolkit requirements.

5. Documentation Management: Management of design and toolkit documentation.

6. Environment Control: Management of the user and toolkit operating environment during a design session.

7. Data Archival: Management of the movement of temporary design information to the toolkit permanent design storage area.

8. Data Access Control: Control of user access to the toolkit, its components, and archived data.

9. Data Security: Management of user's ability to modify toolkit data.

10. Data Storage Maintenance: Control of the space and method used to store temporary and permanent data.

11. Report Generation: Generation of reports concerning toolkit structure, utilization, and capability.

12. Design Standards Management: Management of the structure and documentation of temporary and permanent design information.

13. Design Library Management: Management of the permanent design cells used by toolkit components.

14. Design Schedule Management: Scheduling aids for tracking user-established design milestones and for toolkit maintenance activities.

Design Tool Drawers. The toolkit must include drawers containing the necessary tools to accomplish the general categories of design tasks listed below.

1. Architectural: System level design considerations.

2. Logic: Translation of system design into logic building blocks.

3. Floorplan: Functional partitioning of IC surface.

4. Layout: Translation of logic blocks into devices.

5. Interconnection: Connection of devices.

6. Topological Analysis: Analysis of device layout and interconnection scheme to ensure compatibility with fabrication techniques.

7. Extraction: Extracting device information for use by other analysis tools.

8. Timing Analysis: Analysis of on-chip signal timing relationships.

9. Electrical Analysis: Analysis of individual device electrical characteristics as well as system level electrical characteristics.

10. Mask Data Generation: Conversion of layout and interconnection data to data usable by automated fabrication equipment.

11. Design Verification: Verification at each design level that the implementation meets the design specification at the previous level.

12. Design Presentation: Presentation of the design in a format (usually graphical) easily interpreted visually.

13. Utilities: Accessory tools needed to accomplish minor design tasks.

Information Storage Drawers. The toolkit must include drawers for the temporary and permanent storage of information in the categories listed below.

1. Toolkit and component tool documentation: Includes source code, data and control flow diagrams, textual descriptions, and learning aids.

2. Design libraries: These libraries contain previously designed devices and circuits (often called "cells") which are common to many system architectures. These cells are intended to be incorporated into a system design during the layout step.

3. Toolkit utilization statistics and history.

4. Design schedules: User schedules and toolkit modification schedules.

5. Design data: Data produced by tools at any design level which represents that level's implementation of the design specification.

6. Design documentation: Documentation supporting the design data.

7. Auxiliary data required by component tools: Data needed for operating system interface, tool-to-tool interconnections, etc.

8. Auxiliary data required by toolkit custodian: Operating system parameters, user interface data, data access information, etc.

A Typical Design Session with an Integrated CAD Toolkit

The functional organization of the integrated CAD toolkit described above may be illustrated by outlining a typical design session which uses the toolkit.

One effective way to visualize the design process is to consider the toolkit drawers as distinct magnetic disk storage areas and to treat any toolkit component as an executable computer program. Hence, any reference to "opening a

drawer" signifies an action to access that drawer's storage area and any reference to an action on the part of the custodian signifies the execution of a program residing in one of the particular "drawers".

The sequence of events expected to occur in a "typical" VLSI design session are presented below.

1. The user requests toolkit access from the custodian.

2. The custodian unlocks the toolkit and assumes control.

3. The temporary and permanent information storage drawers are opened.

4. A temporary toolkit session log is created and placed in the temporary information storage drawer. All further custodian and toolkit activities will be recorded in this log.

5. The data access drawer is opened. All further requests for access to tools or data are processed via this drawer of tools.

6. The user's access authorizations are established.

7. If the user is not authorized access to the toolkit then the toolkit is closed and locked (see 16 below) otherwise the session continues.

8. The custodian queries the user to obtain the information necessary to meet the user's requirements for this session.

9. The resource management drawer is opened.

10. The custodian determines whether or not the necessary design resources are available. If the resources are not available the user is given the opportunity to place his request in a toolkit queue for processing at a later time (determined by the custodian). If the user elects to stop working the toolkit is closed and locked (see 16 below).

11. If the necessary design resources are available then the custodian opens the environment con-

trol and data security drawers. The components of these two drawers will be used to: 1) set the resource environment parameters to match those required to interface with the user's external environment and 2) to identify the data which may be used and modified by the user during the session.

12. The data translation drawer is opened. The components of this drawer will be used to process user-specified input data and to process data generated by toolkit components during the completion of user-specified tasks.

13. At this point the custodian opens the appropriate design tool drawers and executes the tools necessary to accomplish the user-specified tasks. During this stage other design management drawers may be opened and closed as necessary to ensure proper management of the design and toolkit.

14. The toolkit components are executed until the user-specified tasks are completed or until the custodian identifies a problem.

15. If a problem occurs, the custodian will identify the problem and suggest a course of action for the user. The toolkit is then closed and locked (see below).

16. When the custodian closes and locks the toolkit the user session history is concluded and incorporated into the toolkit history (maintained in the permanent information storage drawer). All executing toolkit design tools are brought to graceful terminations. The temporary information storage drawer is emptied either by discarding the information stored there or by moving the information to permanent storage if appropriate. All toolkit drawers are then closed. If a condition has occurred during this session which warrants the attention of the person(s) responsible for maintaining the toolkit, appropriate messages are generated (sent) to notify the responsible person(s). The toolkit is locked when the custodian completes execution.

Summary

The first step toward the characterization of a useful, complete, and maintainable VLSI CAD toolkit has been com-

pleted. The proposed VLSI CAD toolkit was partitioned into three sections of drawers: design management tools, design tools, and information storage drawers. A custodian was identified and given responsibility for the control and management of the toolkit. The properties and characteristics of the toolkit, toolkit components, and the toolkit drawers were generally described and were illustrated by outlining a typical VLSI design session. This general depiction of the VLSI CAD toolkit establishes the basis for the development of a more specific toolkit characterization.

A more specific definition of the VLSI CAD toolkit will be presented in the next chapter. Subsequent chapters will draw heavily on that detailed description for the formulation of the weighting metrics and development of the technique that will be used to methodically develop a VLSI CAD toolkit by integrating individual CAD tools.

IV. Specific Characteristics of a VLSI CAD Toolkit

An integrated VLSI CAD toolkit has been described as a collection of drawers of tools managed by a single custodian. The general properties of a VLSI CAD toolkit and the toolkit components have been discussed.

This chapter completes the characterization of the toolkit by describing the specific characteristics and capabilities that the custodian and each toolkit drawer must possess. This discussion is more detailed than that in the previous chapter however, the level of detail has been purposely limited to retain applicability of this description to a wide selection of automated design resources.

Detailed Custodian Specifications. The toolkit custodian must possess the characteristics and capabilities described below.

User Interface. The custodian must be the only user interface to the toolkit. The user interface must provide:

1. The capability to easily select toolkit components at any design level.

2. On-line help and aid to the user at each decision point.

3. Meaningful feedback concerning user entries.

4. Logical, syntactical, and physical error detection as well as the ability to recover from errors.

5. The capability to select an interface display that matches user equipment.

6. Full access to user-selectable options in com-

ponent tools.

7. The ability to identify user-selectable data required by component tools.

8. An interface that is simple enough to service very inexperienced users and flexible enough to meet the demands of expert users.

9. Selectable levels of input assistance.

10. Meaningful feedback during the execution of toolkit components.

11. Command names which may be customized for user convenience.

12. The ability to identify a file of commands which are to be executed as though they were real-time user inputs.

13. The ability to tailor output messages to include user specified data.

14. Verification that the data and tools necessary to fulfill user requests are available in the toolkit.

Access Control. The custodian must provide absolute control of access to the toolkit, the toolkit drawers, and the drawer components. Specifically, the custodian must:

1. Control access to the toolkit, individual drawers, and individual drawer components on a user-to-user basis.

2. Verify a potential user's access authorizations in a manner that does not reveal the method used to determine access nor the access authorizations of other toolkit users.

3. Provide an audit trail of repeated attempts to access the toolkit by an unauthorized user.

4. Maintain a history of all toolkit sessions.

Information Control. The toolkit custodian must control the flow of information into, within, and out of the toolkit. This task is the most important of all custodian

responsibilities. The information management tasks shown below must be included:

1. Input data must be analyzed to ensure that it is in a format which is compatible with the toolkit formats or in a format which is acceptable to the data translation tools available in the toolkit. Improperly formatted data must be rejected.

2. All user input data must be treated as temporary or transient data. User requests to add, delete, or modify permanent toolkit data must be stored by the custodian for review by the person(s) responsible for the toolkit configuration.

3. Once data has been moved to permanent storage, the custodian will prevent modification of the data by any user in subsequent design sessions.

4. Once acquired, user input data will be internally controlled and translated as needed for tool use.

5. Access to any and all temporary or permanent data, either by the user or by a resident tool will be controlled by the custodian.

6. All data output during a design session will be delivered via the custodian.

<u>Resource Management</u>. The custodian must provide continuous management of design resources during any design session. The custodian must be responsive to resource needs of other system users and is responsible for the resource management tasks described below.

1. Controlling a queueing system which will provide user access to toolkit resources on a priority basis.

2. Management of temporary and permanent disk storage space required for toolkit execution and maintenance.

3. Management of toolkit-generated CPU loads.

4. Management of toolkit-generated peripheral

device loads.

5. Management of toolkit-generated CPU temporary memory storage loads.

<u>Detailed Design Tool Drawer Specifications</u>. The toolkit custodian manages three collections of toolkit drawers. One set of drawers, the design tool drawers, are used to perform the actual design. The design tool drawers are described below.

<u>Architectural Drawer</u>. The architectural drawer must contain tools which enable the accomplishment of the following tasks:

1. Capture of the system functional specification.

2. Creation and capture of a hierarchical design at the system level. Captured design data must include inputs, outputs, flow of data, and flow of control for the system and each subsystem.

3. Storage of captured design in a format compatible with the toolkit data translation tools.

<u>Logic Drawer</u>. The logic drawer must contain tools which enable the accomplishment of the following tasks:

1. Translation of the system functional specification into logic building blocks which implement the desired system architecture and which lend themselves to functional verification via automated methods.

2. Minimization of the logic gates within a logic building block.

3. Synthesis of logic building blocks into the highest level blocks compatible with the system specification.

4. Storage of logic block construction data in a format compatible with the toolkit data translation tools.

Floorplan Drawer. The floorplan drawer must contain the tools necessary to accomplish the following tasks [16]:

1. Estimation of total chip area required by each logic building block.

2. Estimation of the total chip area required by the interconnection wiring.

3. Estimation of the total wire length required to interconnect logic blocks.

4. Estimation of the total wire length required by the design.

5. Estimation of the total area required to layout the design.

6. Assistance with the placement of logic blocks to minimize surface area requirements and wire lengths.

7. Assistance in the placement and ordering of pads.

8. Assistance with the placement of power and ground busses.

9. Storage of the floorplan data in a format compatible with the toolkit design presentation tools.

10. Accept logic block data either directly or via the toolkit data translation tools.

Layout Drawer. The layout drawer must contain the tools necessary to accomplish the following tasks:

1. The manual specification and single or iterative placement of individual wires, individual devices, and multiple devices (logic blocks or cells) through the use of symbols and/or a high-level "language".

2. Automatic placement of pre-designed logic blocks or cells.

3. Storage of the layout data in a format compatible with the toolkit data translation tools.

Interconnection Drawer. The interconnection drawer

must contain the tools necessary to accomplish the following tasks:

1. Automatic connection of signal, power, and ground paths between devices and logic blocks.

2. Automatic minimization of interconnection wire lengths.

3. Assistance in the relocation of devices and logic blocks to effect minimal wiring lengths and surface area requirements.

4. Storage of the interconnection data in a format that is compatible with the toolkit data translation tools.

Topological Analysis Drawer. The topological analysis drawer must contain the tools necessary to accomplish the following tasks:

1. Analysis of device,logic block and connection wiring layout to ensure there are no violations of an externally specified set of topological design rules.

2. Storage of the analysis data in a format that is compatible with the toolkit data translation tools.

Extraction Drawer. The extraction drawer must contain the tools necessary to accomplish the following tasks:

1. Extract device, logic block, and wiring descriptions that are compatible with the toolkit timing analysis and electrical analysis tools.

Timing Analysis Drawer. The timing analysis drawer must contain the tools necessary to accomplish the following tasks:

1. Perform a static analysis of on-chip signal relationships given an external description of the timing signals.

2. Estimate signal propagation delays by device, logic block, and interconnection.

3. Estimate point-to-point signal propagation delays through multiple devices, logic blocks, and interconnections.

4. Store the analysis and estimation data in a format that is compatible with the toolkit data translation tools.

<u>Electrical Analysis Drawer</u>. The electrical analysis drawer must contain the tools necessary to accomplish the following tasks:

1. Determine the average, RMS, and peak power consumption of the chip.

2. Determine the voltage and phase relationships between nodes on the chip.

3. Determine the capacitance between any two nodes on the chip.

4. Determine the frequency response between any two nodes on the chip.

5. Determine the rise and fall times of s signal at any node on the chip.

6. Store the analysis data in a format that is compatible with the toolkit data translation tools.

<u>Mask Data Generation Drawer</u>. The mask data generation drawer must contain the tools necessary to convert the design description from a locally stored format to a format compatible with automated mask generation equipment.

<u>Design Verification Drawer</u>. The design verification drawer must contain the tools necessary to accomplish the following tasks:

1. Verify that the system architecture meets all system functional specifications.

2. Verify that the logic design implements the system functional specification.

3. Verify that the layout implements the logic

design.

4. Verify that the layout meets the timing requirements established in the system specification.

5. Verify that the layout meets the electrical requirements established in the system specification.

6. Verify that the mask generation data implements the desired layout.

7. Verify that the fabricated device implements the desired system specification (must be used in conjunction with external hardware).

<u>Design Presentation Drawer</u>. The design presentation drawer must contain the tools necessary to accomplish the following tasks:

1. Provide a real-time graphical representation of the design at any design level.

2. Provide a "hard-copy" representation of the design at any design level.

3. Provide a representation of the design at any design level which is transportable to compatible computers.

<u>Utility Drawer</u>. The utility drawer may contain tools to accomplish the types of tasks described below.

1. Scale a completed device, cell, or design.

<u>Detailed Design Management Drawer Specifications</u> The custodian manages past, in-progress, and planned designs through the use of the design management drawers. The design management drawers are described below.

<u>Configuration Drawer</u>. The configuration drawer must contain the tools necessary to: 1) manage the additions, deletions, and modifications to the toolkit drawers and 2)

manage modifications to the toolkit custodian.

Data Translation Drawer. The data translation drawer is one of the most important drawers in the toolkit. All translation and reformatting of data used by the toolkit is performed by the tools in this drawer. The following translation capabilities must be provided:

1. Translation of design data input by the user into a format compatible with the toolkit databases and toolkit component tools.

2. Translation of tool-generated data to permit inter-drawer data transfer.

3. Translation of tool-generated data into formats compatible with toolkit databases.

4. Translation of output data into a format selected by the user.

Resource Management Drawer. The resource management drawer must contain the tools necessary to ensure effective custodial management of design resources. The following resource management tasks must be included:

1. Analysis of temporary and permanent storage capacities to ensure user and tool demands do not exceed toolkit allocations.

2. Analysis of CPU usage requirements to ensure that user and tool demands will not exceed toolkit CPU allocations.

3. Analysis of temporary and permanent storage allocations to remove redundant data.

4. Analysis of toolkit usage on a user basis to ensure usage does not exceed the allocations for that user.

5. Analysis of existing resources to ensure user and tool demands can be met prior to any attempt to meet the demands.

6. Maintenance of a "queue" to ensure all users

have equitable access to toolkit resources.

Data Analysis Drawer. The data analysis drawer must contain the tools necessary to provide the custodian and users with statistics concerning toolkit and design data. The scope of the analysis capability must be sufficient for the custodian to properly assess the status of toolkit data in order to fulfill management functions. The scope of the analysis capability must also be sufficient to permit users to make decisions at each design level.

Documentation Management Drawer. The tools in this drawer must provide the custodian with absolute control over the documentation of designs, drawers, and tools which are permanent residents of the toolkit.

Environment Control Drawer. The environment control drawer must contain the tools necessary to maintain all toolkit interfaces. This drawer must include:

1. Toolkit-to-operating system interface information and management tools.

2. Toolkit-to-peripheral interface information and management tools.

3. Inter-drawer interface information and management tools.

Data Archival Drawer. This drawer must contain the tools necessary to provide the custodian with absolute control over the temporary design data that is incorporated into the permanent toolkit database.

Data Access Control Drawer. This drawer must provide the custodian with the tools necessary to exercise absolute control, on an individual user basis, to user access to

toolkit data and tool drawers.

Data Security Drawer. This drawer must contain the tools necessary for the custodian to exercise absolute control, on an individual user basis, over the user's ability to modify toolkit data either directly, or indirectly through the use of tools.

Data Storage Management Drawer. This toolkit drawer must contain the tools necessary to manage size and format of the temporary and permanent toolkit design databases. The capability to selectively extract data from the databases must be provided to the custodian and to authorized users.

Report Generation Drawer. This drawer must contain the tools necessary to generate reports concerning toolkit usage and contents.

Standards Management Drawer. This drawer must contain the tools necessary to ensure user input data, tool-generated data, data resident in toolkit databases, and toolkit output data meet a predetermined set of standards concerning format and content.

Library Management Drawer. This drawer must contain the tools necessary to manage toolkit libraries of cells which are intended to be incorporated into future designs.

Schedule Management Drawer. This toolkit drawer must contain tools sufficient to permit the establishment and tracking of user-created design schedules and custodian-generated schedules. Schedule management must be provided

for single-user designs, multi-user designs, and long and short term custodian schedules.

Detailed Information Management Drawer Specifications The third set of drawers used by the toolkit custodian are the information management drawers. These drawers are used to store the temporary and permanent information necessary to accomplish and manage VLSI designs. A description of the information management drawers is provided below.

Temporary Storage Drawer. This toolkit drawer will be used for the temporary storage of data required by the custodian or by any toolkit drawer. This drawer will not be accessible for storage by any toolkit user and will be emptied when the custodian closes the toolkit.

Permanent Storage Drawer. This drawer will be used for the permanent storage of data required by the custodian or any toolkit drawer.

Summary

This chapter concludes the characterization of a useful, complete, maintainable, integrated VLSI CAD toolkit. The level of detail in this characterization has been purposely limited to retain applicability of this description to a wide spectrum of design resources.

The need for a toolkit of this nature has been established. This characterization of the toolkit as a collection of drawers managed by a custodian will provide the basis for development of a method to integrate individual

CAD tools into such a toolkit. Subsequent chapters will detail the integration method, categorize the CAD tools available at AFIT, and apply the integration method to the AFIT tools.

V. A Systematic Method for Integrating VLSI CAD Tools

Introduction

Discussions in previous chapters have established the importance of integrated CAD tools to the success of future VLSI design efforts. A "toolkit" of integrated VLSI CAD tools was described as a collection of drawers managed by a "custodian". The general and specific characteristics of a useful, complete, and maintainable CAD toolkit were presented. The definition and description of an integrated VLSI CAD toolkit represent the bulk of the effort necessary to actually achieve such a toolkit. What remains is the need for a systematic method to collect, develop and combine CAD tools in such a manner as to obtain the desired toolkit characteristics. The goal of this chapter is to present that systematic method.

Philosophy of This Integration Method

This chapter will present a general, reasonably flexible method suitable for constructing a useful, complete, and maintainable VLSI CAD toolkit. It has been stated many times throughout the course of this thesis that emphasis will be placed upon academic needs. That emphasis surfaces during this development as a departure from the rigid pursuit of a fully integrated toolkit.

Any pragmatic approach to the systematic integration of CAD tools for academic purposes must permit the establishment of a lesser goal than the implementation of a useful,

complete, and maintainable toolkit. Most academic institutions simply do not possess, and cannot rapidly obtain, the resources required to implement a fully-integrated toolkit. The lack of resources is compounded by the urgent need for graduates skilled in VLSI design methods and tools. The approach presented here is designed to meet the academic short-term goal, the rapid construction of a usable, partially-integrated toolkit, as well as the long-term goal of implementing a fully-integrated toolkit.

The goal of obtaining a useful, complete, and maintainable toolkit must never be ignored but may be delayed with the reality of academic requirements and resources in mind. The interests of the long term toolkit goal are preserved by the requirement that any components of a "lesser" toolkit must be subsets of the set needed to implement a fully integrated VLSI CAD toolkit.

The integration technique presented here is meant to be a meaningful guideline for CAD tool integration. This technique is neither absolute nor infallible and may not be suitable for each and every CAD tool collection. The method permits, in fact requires, tailoring and optimization by each toolkit architect. The reader should leave this chapter with a picture of a useful integration procedure that is receptive to future refinement and expansion.

<u>The General Integration Methodology</u>

This CAD tool integration method is similar to any design effort in that it is an iterative process with inher-

ent feedback. There are nine general tasks that must be accomplished sequentially during the integration process. The first eight tasks must be iterated until a toolkit of acceptable "quality" is realized. The ninth task, implementation of the toolkit control mechanism (custodian) completes the integration. The tasks are:

1. Identify a set of characteristics which completely described the desired toolkit.

2. Define a set of metrics sufficient to measure the presence of desired toolkit characteristics.

3. Perform the high-level design of the control mechanism that will be used for toolkit management and operation (the custodian).

4. Collect existing CAD tools and position them in their logical places (drawers) within the toolkit structure.

5. Identify the drawers in the toolkit structure which have no components.

6. Complete the toolkit structure by : 1) obtaining new tools, 2) creating new tools, or 3) modifying existing tools.

7. Evaluate the toolkit quality as a function of the previously developed metrics.

8. Remove or modify those components which fail the evaluation.

9. Complete the custodian design and implement the custodian.

Toolkit Integration Mechanics

The mechanics of CAD tool integration are detailed below. These mechanical steps, when tailored for application-specific requirements, will lead to a useful, complete, and maintainable VLSI CAD toolkit.

Step 1: Identification of Desired Characteristics. The characteristics established to describe a desired toolkit and any toolkit component must be non-overlapping, reasonably exhaustive, and sufficient to permit evaluation [5]. The characteristics selected must encompass those necessary to ensure the toolkit will be useful, complete, and maintainable. The toolkit's architect and users must realize at the outset of an integration effort that this step is perhaps the most difficult step of all. The major problem is that many of the individual characteristics of quality are in conflict: for example, added efficiency is obtained at the expense of portability, accuracy, understandability, and maintainability; added accuracy decreases portability due to dependence upon the computer word size; conciseness often reduces understandability [5]. Users often have difficulty in quantifying their preferences in such conflict situations [5].

The high level characteristics of an integrated VLSI CAD toolkit have been described in previous chapters of this thesis. Much of the motivation for the effort spent in the development of those characteristics was the desire to provide a useful baseline intended to reduce the effort spent at this integration step by those who follow. What remains is the need for the toolkit architect to define the characteristics of the individual drawer components.

Step 2: Definition of the Evaluation Metrics. An understanding of the general characteristics of evaluation

metrics is necessary if the rapid development of useful metrics is to be accomplished. Several important metric characteristics were summarized in a recent study concerning the evaluation of software quality [5]. Among those characteristics were:

1. There is no single metric which can give a universally useful rating of software quality.

2. One is generally interested in where and how rather than how often a product is deficient. Therefore the most valuable metrics are those which flag deficiencies and anomalies rather than just producing numbers.

3. For all simple quantitative metric formulas, it is easy to find counter-examples which challenge their credibility.

4. The software field is developing too rapidly to establish metrics in some areas.

5. At best, a prospective (metric) user could receive a useful rating system by furnishing the quality rating system with a thorough set of checklists and priorities.

6. Since metrics are not exhaustive, the overall rating obtained using them would be more suggestive rather than conclusive or prescriptive.

Some readers may interpret the metric characteristics presented above as indicators of the uselessness of software metrics. Metrics, with all their limitations, provide the only means available to practically and methodically evaluate software. A careful consideration of the metric characteristics listed above should help explain and prevent much of the indecisiveness associated with attempts to quantitatively identify what is "best". The metric definition technique presented below is intended to identify software products (tools) that are "probably most suitable".

Specific CAD Tool Metric Characteristics Each metric developed to measure a toolkit component must possess three fundamental characteristics: 1) The metric must reflect the essentiality of the characteristic to the successful integration of the toolkit (essentiality value : EV), 2) The metric must reflect the "level" to which the desired characteristic is present while simultaneously signalling the presence of anomalies or deficiencies (characteristic level: CL), and 3) The metric must be easy to compute. These characteristics provide the guidelines necessary to construct the metric scales and formulas that will be used to compute numeric indicators of the "quality" of a toolkit component.

Metric Scales The scales shown below provide a range of values sufficient to identify the essentiality of a toolkit component and to indicate the degree to which a component possesses a desired characteristic.

The scale shown below provides a measure of the essentiality of a characteristic to the success of toolkit integration. A non-linear, increasing scale will place emphasis on the more essential characteristics. Although many other scales may have been constructed, it is believed that this scale is sufficient for the purposes of this thesis.

Essentiality Value (EV)	Essentiality of Characteristic To Successful Integration
32	Extremely important to success (Certain failure if missing)

16	Important to success (Probable failure if missing)
8	Fairly important to success (Possible failure if missing)
2	Has some impact upon success (No predictable impact upon success if missing but less than optimum integration is probable)
1	Slight impact upon success (No predictable impact upon success if missing but less than optimum integration is certain)

The scale shown below provides a measure of the presence of a desired characteristic. The negative values were included to provide a "penalty" for anomalities and deficiencies.

Characteristic Level (CL)	Degree to Which Characteristic is Present in this Component
3	The desired characteristic is entirely present.
2	The desired characteristic is mostly present.
1	The desired characteristic is somewhat present.
0	The desired characteristic is completely absent.
-1	Characteristics are present which are somewhat detrimental
-2	Characteristics are present which are very detrimental
-3	Characteristics are present which are unacceptable

The Metric Formulas The formulas presented below

are designed to compute numeric values which will be used as indicators of the relative quality of the toolkit, any toolkit drawer, and each of the drawer components.

The overall quality of any component, drawer, or the toolkit is defined by these formulas. The formulas will be used during the evaluation step detailed in subsequent paragraphs.

Drawer Component Evaluation Formula:

$$DCQ = \frac{\sum_{1}^{i} [\ EV(i) \times CL(i)\ ]}{n} \qquad \text{(EQ 1)}$$

Where :

DCQ = Drawer Component Quality (overall)
EV = Essentiality Value of characteristic i
CL = Characteristic Level of characteristic i
n = Number of characteristics being evaluated

Drawer Evaluation Formula:

$$DQ = \frac{\sum_{1}^{i} [\ EV(i) \times DCQ(i)\ ]}{n} \qquad \text{(EQ 2)}$$

Where :

DQ = Drawer Quality (overall)
DCQ = Drawer Component Quality of drawer component i

EV = Essentiality Value of drawer component i
n = Number of drawer components

Toolkit Evaluation Formula:

$$TQ = \frac{\sum_{1}^{i} [\ EV(i) \times DQ(i)\ ]}{n} \qquad \text{(EQ 3)}$$

Where :

TQ = Toolkit Quality (overall)
DQ = Drawer Quality of drawer i
EV = Essentiality Value of drawer i
n = Number of drawers

<u>Step 3: High-Level Custodian Design</u> The design of the toolkit custodian must be accomplished in a hierarchical fashion. At this level the design is concerned with the custodian architecture and with the flow of control and data within the toolkit. The design should be documented using a combination of text and charts which clearly indicate the flow of data and control. Use of the Structured Analysis and Design Technique (SADT) technique (developed by SofTech Corp.) is highly recommended.

Care must be taken during this level of design to ensure the design is implementation independent. The design documentation should be sufficiently detailed to enable the toolkit architect to clearly illustrate the toolkit structure to potential users and those tasked to implement the toolkit.

The custodian design should be performed with a

constant awareness of the desired characteristics and continuous reference to the evaluation metrics which have been previously established for those characteristics.

A representative high level custodian design, based upon the characteristics described in previous chapters, has been performed and is included as Appendix A to this thesis.

Step 4: Collect Existing CAD Tools At this step in the integration process, all existing CAD tools which appear to have a useful place in the VLSI CAD toolkit are collected and placed in appropriate drawers. It is possible that a versatile CAD tool may fit into one or more drawers and should be so placed. Caution should be observed when evaluating a *single tool appearing in multiple drawers* to ensure that only those features of the tool which are pertinent to its CURRENT drawer are evaluated. The best approach is to place a tool in a drawer even though its usefulness is questionable. The tool will be eliminated during evaluation if it is of no real value.

An example of this step has been performed using the CAD tools currently available to AFIT. The distribution is provided as appendix B to this thesis.

Step 5: Identify Empty Toolkit Drawers Identification of empty toolkit drawers is the easiest step in the integration process. Filling the empty drawers may be one of the more time-consuming steps.

Step 6: Complete the Toolkit Structure During

this step, empty toolkit drawers are filled in one of the following manners: 1) Obtaining new tools, 2) Creating new tools, or 3) Modifying existing tools.

This is the first decision point concerning whether or not to pursue the implementation of a fully integrated toolkit. It is believed that most academic institutions will not have sufficient resources to complete the toolkit the first time integration is performed. At this point, the goal may be reduced to creating a toolkit which is immediately usable. If a fully integrated toolkit is not desired or possible immediately, then complete only those drawers necessary to make the toolkit usable. The empty drawers should be marked for completion at a later date (perhaps during subsequent courses or thesis activity). The next decision point of similar consequence occurs during the evaluation step.

If a fully integrated toolkit is immediately required then the drawers must be filled before proceeding. The first step in the process to complete the drawers should be a careful re-examination of the existing tools. Perhaps a tool possessing the necessary characteristics has been overlooked. Check all other drawer components. Local creation of new tools has a high probability of success (since the desired characteristics and evaluation criteria are known). However, software creation is a time-consuming and laborious process requiring proper resources. Obtaining (purchasing) additional tools is the final option. If pos-

sible, purchase contracts should include the desired characteristics and the evaluation metrics as a formal part of the component specification.

Step 7: Toolkit Evaluation The toolkit evaluation is a critical step in the integration process. The evaluation should be accomplished by those responsible for the toolkit implementation, those responsible for toolkit maintenance and by as many of the potential users as possible. The evaluation should be performed under the guidance of the single person or group that has the overall responsibility for the toolkit. The person or group that has the overall toolkit responsibility must be the final authority in all decisions concerning the toolkit.

The evaluation process is a six-stage process consisting of:

1. Assigning essentiality values (EV) to each desired characteristic of the drawer components, drawers, and the toolkit.

2. Determining characteristic level values (CL) for each drawer component.

3. Computing the drawer component quality value (DCQ) for each of the drawers.

4. Computing the drawer quality value (DQ) for each of the drawers.

5. Computing the toolkit quality value (TQ) for the toolkit.

6. Determining if the toolkit quality value (TQ) is acceptable.

The second decision point concerning whether or not to pursue a fully integrated toolkit occurs when the

toolkit quality value is examined. Acceptance of a "less-than-optimum" toolkit is an alternative available to those who wish to have a usable toolkit immediately available even though the toolkit is not complete or maintainable. A realistic view of the current state of most academic institutions indicates this may be the case.

If the decision is made to accept a toolkit which is not fully integrated, then a commitment must be made to vigorously pursue complete integration. Although a partially integrated, controlled set of CAD tools is preferable to unorganized tools, the goals stated early in this thesis cannot be achieved without a useful, complete, and maintainable VLSI CAD toolkit.

An example of the evaluation process, an evaluation of a sample custodian constructed for use at AFIT (contained in appendix C), is included as appendix D to this thesis.

<u>Step 8: Remove or Modify Components</u> The decision to remove or modify those components which fail evaluation must be the responsibility of those tasked to implement the toolkit. Removal of components necessitates a reiteration of the integration process. Modification of a component or components necessitates only a reiteration of the evaluation step.

<u>Step 9: Completion of the Custodian</u> Completion of the custodian is necessarily the last step in the integration process. At this point in the process all other

toolkit components have been favorably evaluated. The custodian will provide the control and cohesion necessary to make the toolkit function as a single entity.

During this step the high-level design of the custodian is reviewed and corrected to correlate with the toolkit structure remaining after the last evaluation. Next, the high-level custodian design is implemented using an available language. Preference should be given to high-level languages and to languages which enforce structured programming techniques since they are more likely to enable the creation of maintainable software.

A sample custodian, written in the UNIX "C Shell" language, is provided as appendix C to this thesis.

Summary

The mechanics of CAD tool integration have been presented. The nine-step method has been proposed which represents a general but flexible way to implement the type of toolkit specified in previous chapters. A set of metrics has been presented which will provide a satisfactory way to systematically rate the toolkit and its components. The reader was shown the decision points where the integration method permits the user to postpone full integration to obtain a partially-integrated but usable collection of tools.

In the interest of improving the readability of this work, several examples, intended to illustrate the integra-

tion process steps have been moved to the appendices. These examples are essential elements of this work and must not be overlooked or excluded if the reader is to obtain a full understanding of the VLSI CAD tool integration process.

The next chapter will provide a discussion of the work done to create the examples contained in the appendices and should serve as a first-level evaluation of the validity of the methods suggested herein. The final chapter will summarize this author's work and will include recommendations for future work.

VI. Discussion of the Prototype Implementation

Introduction

The inability to optimize time-limited, VLSI-oriented academic curricula has been identified as a problem which seriously hinders the education and production of the types of engineers that will be required to meet future VLSI design challenges. The background surrounding this problem has been presented in Chapters 1 and 2 of this thesis. The availability of an integrated, technically-complete, academically-oriented VLSI CAD toolkit has been identified as a possible solution to the optimization problem. Chapters 3 and 4 have presented the general and specific characteristics of the required toolkit. Chapter 5 detailed a systematic method and a set of *CAD tool evaluation metrics* proposed as sufficient to implement an integrated VLSI CAD toolkit.

An effort has been made to evaluate the proposed toolkit structure and the integration methodology through the implementation of a prototype toolkit structure and its custodian. The details of the prototype implementation have been provided as appendices A, B, C, and D. This chapter presents a discussion of the prototype effort.

The appendices have been provided to serve two purposes: 1) to illustrate the proposed toolkit structure and integration methodology and 2) to provide enough information to enable a rudimentary evaluation of the utility of the toolkit structure and the integration methodology. The appendices will be discussed in their order of appearance

which has been arranged to follow the actual sequence of steps used during the construction of the prototype toolkit.

High-Level Design of the Toolkit Custodian

The high-level design of the toolkit custodian is presented as appendix A.

Goal. The goal of the work presented in this appendix was the illustration of the technique and documentation required during the functional requirements analysis phase of any high-level design of a VLSI CAD toolkit custodian program.

Contents. This appendix contains the products of a partial, high-level design of a prototype toolkit custodian. The design was accomplished using the Structured Analysis and Design Technique (SADT). The SADT technique is the recommended design technique and the products provided in this appendix illustrate those required at this design level. The products provided include: 1) SADT diagrams and the necessary accompanying text pages, 2) a node index, and 3) a data dictionary. The reader is alerted to an unusual page numbering method employed in this appendix. The pages containing the text associated with the SADT diagrams have been numbered on the front while the text has been provided on the reverse side of the page. This method was used to ensure the text and SADT diagrams would face each other (required by SADT documentation standards).

Comments. It should be clear that the high-level de-

sign of a program does not begin and end with the products provided in this appendix. The design begins with a clear statement of requirements which are completely understood and agreed upon by the designer and the originators of the requirements. The requirements for this design were extracted from the VLSI CAD toolkit characterization developed in Chapters 3 and 4 of this document.

The SADT technique is used to transform user requirements into a high-level design which clearly depicts the design structure and which is easily analyzed and understood by both the user and the designer. The combination of graphical and textual materials in the design documentation appear to be very effective when used during design reviews.

The design documented in Appendix A is a partial, hierarchical design developed only a few levels deep. Normally, the design would be developed to a level of detail just above that which requires implementation-dependent information. The reader should observe the hierarchical nature of the design and the utility of the SADT method in presenting both the form and content of the design.

The successful completion of this design phase is a critical milestone in the toolkit integration process. It is at this level that the potential user has the best opportunity to ensure the toolkit structure will satisfy requirements. From the designer/integrator perspective, this design phase affords the best opportunity to resolve requirement and design conflicts. The SADT documentation

technique also forces the designer to develop and organize a logical design approach. The completion of this design phase does not signal the end of the design task. A proper approach to the design of the custodian, and any other piece of complex software, should include a systems design phase and a detailed design phase.

Although the systems and detailed design phases were incorporated in the prototype custodian design, time did not permit their formal documentation. When the high-level design was finished, a prototype toolkit framework and its drawer structure were implemented.

Implementation of Toolkit Framework and Drawer Structure

A prototype implementation of the proposed toolkit structure and its "drawers" is presented as Appendix B.

Goal. The goals of this portion of the prototype effort were twofold: 1) to illustrate one method of actually implementing the proposed toolkit and its "drawers" and 2) to evaluate the practicality of the proposed structure.

Contents. Appendix B documents the prototype toolkit structure as it was implemented using the hierarchical directory feature of the AFIT VAX 11/780 UNIX (Bell Laboratories) operating system. The toolkit structure is documented in four parts: 1) a description of the mapping that was employed to map the toolkit drawer names to UNIX directory names, 2) a graphical depiction of the toolkit structure using the UNIX directory names assigned to the toolkit drawers, 3) the source code for the UNIX command language

("C Shell") program which was used to created the toolkit structure on the AFIT VAX 11/780 and 4) a detailed listing of a first-effort to distribute existing AFIT VLSI CAD tools within the prototype toolkit structure.

Comments. The hierarchical directory feature of UNIX lends itself nicely to the implementation of the suggested toolkit. The composition and execution of the program to build the directories was very straightforward. The availability of hierarchical directories should not be viewed as a "convenient" and isolated feature available only to this author. The early and continuing emphasis of this thesis has been to create an "academically-oriented" toolkit. The UNIX operating system has found favor with many academic institutions and certainly with those interested in the development of VLSI CAD tools. The hierarchical directory feature is also emerging in the new generations of micro-computer operating systems.

The structure of the "directory-oriented" toolkit is easy to observe and understand. Modifications to the structure should be simple and easily accomplished.

The distribution of existing CAD tools proved to be a major task. The listing provided in the latter part of this appendix clearly demonstrates the magnitude of the task. The tool distribution effort revealed the need to identify several partitions to the task. The distribution of existing tools must be partitioned into the following subordinate tasks:

1. The tool distribution list should be independently formed by several people familiar with the capabilities and limitations of the CAD tools. This method should ensure correct tool distribution using "corporate memory" and should optimize tool placement through the identification of tools capable of providing multiple functions. Assignment of the distribution task to an individual or a very limited group of people invites failure.

2. Once the tool distribution has been accomplished, the implementation-specific portions of the source code for each tool must be modified to reflect the new "location" of the tool and the new "location" of tools and data required by the tool. This task must be accomplished by people intimately familiar with the tool's source code language, the support computer operating system, and the tool itself. Although the requirements for this step appear to be very demanding, the necessary expertise may be found in most academic institutions.

The distribution of the current AFIT CAD tools has revealed the areas or drawers of the proposed toolkit for which there are no exiting tools. These drawers are: 1) Configuration, 2) Resource Management, 3) Environmental Control, 4) Data Access Control, 5) Data Security, 6) Report Generation, 7) Standards Management, 8) Schedule Management, 9) Architectural, 10) Floorplan, and 11) Interconnection. Several iterations of the distribution process using the technique suggested above will certainly eliminate some of the empty drawers. The lack of components for the Standards Management drawer is of particular significance. The implementation of uniform standards for toolkit components and data will require the development of a database structure for design data (including libraries of data) and the creation of programs sufficient to manage the database.

Observation of the quantity and diversity of the VLSI CAD tools currently available at AFIT supports the suggestion that a method must be implemented to rapidly provide students with easy and effective access to the tools.

A copy of the program used to build the prototype toolkit structure and a copy of the distribution listing have been placed in the directory named "/usr/local/cad/trv" on the AFIT VAX 11/780. The distribution listing was created using the current, complete UNIX path names to the existing tools. The intention was to facilitate the construction of a UNIX "C Shell" program which could use the listing to actually find and move the existing tools to the toolkit structure.

Custodian Source Code

A prototype implementation of the toolkit custodian is presented in Appendix C.

Goals. The goals of this prototype task were: 1) to illustrate the structure of a useful toolkit custodian and 2) to provide a toolkit drawer which could be evaluated using the methodology and metrics developed in this thesis.

Contents. Appendix C contains the source code for a prototype toolkit custodian program written using the UNIX "C Shell" command language. The line numbers are provided for easy reference during this discussion and must be removed prior to attempting to use the code. A copy of the custodian has been placed in the directory named

"/usr/local/cad/trv" on the AFIT VAX 11/780. Also included in the directory are a small number of files which are sufficient to permit a demonstration of the custodian. Throughout the remainder of this section discussions of custodian features will be referenced to the source code by including the appropriate source code line numbers in parentheses.

Comments. The prototype custodian implements most of the structure presented in Appendix A as node "CUS0". The high-level design identifies four major custodian tasks or processes. The prototype incorporation of each of the processes is discussed below.

1. Assign User Access Level (CUS1): This process is implemented in the prototype toolkit through the use of lists stored in disk files. Three lists of UNIX "login names" were created. Each list represents a different level of access to the toolkit. The lists are stored in the "Data Access Control" drawer (36-47). Three levels of access were established with level 1 representing the lowest level and level 3 the highest. Level 3 represents the level of access which would be provided those responsible for toolkit design and maintenance. A user's access level is established early in the execution of the custodian (168-227). The access level is used frequently throughout the custodian during the presentation of menus and during the validation of menu selections (312, 317, 322, 619, 623, 729, 753, 777, 807, 1050). The access level is also used to determine whether or not a user is permitted to execute a particular toolkit component (373-379). Three lists of toolkit components are maintained in the "Data Access Control" drawer. The lists are segregated according to access level. List 1 contains those tools which may be executed by users with access level 1, list 2 contains tools accessible to users with level 2 access, etc. Some user permissions are verified via the operating system (145, 149, 478, 520, 562, 938, 977, 1075, 1097, 1135).

2. Open Toolkit (CUS2): This major process was designed using two subordinate processes: 1) open for authorized user and 2) record unauthorized user access attempt. Both processes were addressed in the prototype custodian. The first step in the "opening" is the disabling of the user's ability to interrupt the custodian program (8-13). The second step is the creation of a temporary log of the session (110-227). During the creation of the session log, each session is assigned a unique session ID composed of the date and the process ID assigned to the current execution of the custodian. Also, the user access lists (same as the access level lists described above) are scanned to see if the current user is authorized access. If the current user's "login name" does not appear in any of the access lists, then the attempted access is recorded in a security log (187-192). The information recorded in the log identifies the user, the time of the security violation, and the session ID. Any attempt to gain access by an unauthorized user results in a message to the user advising him of the steps necessary to gain access authorization. The instructions to the user include the name and location of the person responsible for toolkit management (199-211). The toolkit is gracefully closed following an unauthorized access attempt. If the user is permitted access, then the toolkit remains open, user interrupt capability is restored (228) (jumps to graceful close on interrupt), and the user is presented with the first menu (231).

3. Provide Toolkit (CUS3): This portion of the custodian was designed with four primary processes in mind: 1) obtain user input, 2) analyze user input, 3) respond to user input, and 4) analyze response. Each of the processes above were incorporated in the prototype custodian. The user input is obtained primarily as a response to a menu of options. Six selection menus are provided (231-249, 291-333, 603-630, 811-824, 912-929, 1048-1065). The analysis of a user's response is accomplished through the use of "case" statements with a "default". If an invalid input is detected, the default case statement displays the contents of a "help" file to the user. Other user inputs are obtained as needed. The custodian response to a user input consists of messages to the user, tests for the existence and access permission to user requested data, execution of user requested tools, and log entries. A typical custodian response is shown in the component execution portion of the program (344-444). Finally,

the custodian "analyzes" a component execution response by displaying the status code assigned to the execution by the operating system. The status code is shown to the user and is also stored in the toolkit log (398).

4. Close Toolkit (CUS4): The prototype custodian includes the necessary steps to "gracefully" close the toolkit (1180-1247). The beginning of the "closing" is identified by a program label (close_kit). This label is the re-entry point in the custodian immediately following a user attempt to interrupt custodian execution. The closing process performs 4 steps: 1) Temporary storage space used by this user session is cleared (1228-1232), 2) the toolkit management data is updated (1193-1223), 3) the temporary session log that was created for this user session is appended to the permanent toolkit history log (1226), and 4) the user is notified that the toolkit is closed and also of any processes that remain in execution as a result of this user session (1237-1244).

The custodian program, implemented in the UNIX "C Shell" is rather limited since the implementation language does not permit the use of "structured" programming techniques. The structure and content of the prototype custodian may be converted to a more powerful language (eg. "C"). The prototype custodian is comprehensive enough to demonstrate its flexibility and utility and is complex enough to evaluate using the evaluation methodology and metric proposed as part of this thesis.

Evaluation of the Prototype Custodian

An evaluation of the prototype custodian is presented as appendix D.

Goals. The goals of this evaluation are to illustrate the evaluation of a typical toolkit drawer or drawer component and to test the utility of the evaluation metrics and

formulas.

Contents. Appendix D contains a review of the evaluation metrics and formulas that were developed in Chapter 5. The appendix also contains an evaluation of the prototype custodian described above. The component descriptions and desired characteristics that were used during the evaluation are based upon the custodian characteristics presented in Chapters 3 and 4. In this case, the toolkit drawer being evaluated is the custodian and the drawer components are the desired custodian characteristics. In the general case, the drawer being evaluated could be any toolkit drawer and the components would be the individual contents of the drawer (eg. programs). This evaluation method may also be applied to the drawer components themselves.

Comments. The evaluation presented in this appendix is admittedly biased. A strong effort has been made to reduce the effects of the bias so they have an insignificant effect upon the general outcome of the evaluation. The application of the evaluation metrics to the custodian was relatively simple. Perhaps the simplicity of the evaluation is due to the fact that only a single person was involved and that person was very familiar with the component being evaluated.

The evaluation was conducted in three steps: 1) essentiality values (EV) or "priorities" were assigned to each desired component characteristic, 2) the prototype custodian was examined and a characteristic level (CL) or "presence" level was assigned to each characteristic, 3) the evaluation

formulas were applied.

A summary of the component ratings is presented on page D-19. The summary provides a means of comparison between the maximum attainable ratings (calculated by assigning the characteristic level "3" to each characteristic being rated) and the ratings actually obtained. The results of the evaluation are encouraging. Two useful measures emerged from the evaluation: 1) an absolute measure of the strengths and weaknesses of the component and 2) a relative measure which may be used to evaluate the component against a different component evaluated using the same set of desired characteristics.

The absolute measure of component strengths and weaknesses may be identified by establishing meaningful "cutoff" values within the range of possible rating values. In this case, the following values were selected: 1) a component characteristic was considered missing if the actual rating was zero, 2) a component characteristic was considered to be present but "weak" if the actual rating was less than 50 percent of the maximum possible rating, and 3) a component characteristic was considered to be present and "strong" if the actual rating was greater than 80 percent of the maximum possible rating.

Using the values established above the prototype custodian possesses the strengths and weaknesses listed below.

1. Absent Characteristics:

    a. Selectable levels of user assistance.
    b. Ability to customize command names.

c. Ability to execute commands input from disk files.
d. Ability to tailor output messages to suit the user.
e. Storage of requests by the user to add, delete, or modify the toolkit in the toolkit temporary storage area.

2. Characteristics present but "weak":

   a. On-line help and aid to the user at each decision point.
   b. Input data format analysis.
   c. Treatment of all user input data as temporary data.
   d. Complete control of all data after it has been identified as input data.
   e. Complete control of the parameters used during component execution.

3. Characteristics present and "strong":

   a. Capability to easily select toolkit components.
   b. The user can easily identify user-selectable data needed for component execution.
   c. User access levels are verified in a manner that does not reveal the verification method.
   d. A history is maintained for all toolkit sessions.
   e. The custodian controls access to all drawer components.
   f. The user cannot modify data resident in permanent storage.
   g. The custodian controls the sequence of drawer usage.
   h. The custodian controls the sequence of component execution.

The second, and most useful, measure that emerged during the evaluation was the relative measure of the component. This measure is described in the evaluation as the drawer quality (DQ). This measure identifies the level of success in achieving all the desired drawer components. In this case, the level of success was approximately 67 percent. This value, when compared to other evaluations done

using the same set of desired characteristics, can be used to select the "best" drawer from a collection of similar drawers. Selection could be based upon the drawer with the highest drawer quality.

This evaluation technique shows promise. The technique is simple enough to be used by technical and non-technical users. Further testing of the method is required. Perhaps a useful approach would be to permit several potential users, designers, and maintainers to perform the evaluation. The collection of evaluation data could then be statistically averaged and a generally agreed-upon set of ratings established for the drawer.

Summary

A prototype toolkit and its custodian have been designed, implemented, and evaluated with the intention of illustrating and evaluating the proposed CAD tool integration methodology. The tasks involved in the prototype task have been discussed and a preliminary evaluation of integration methodology has been presented.

Implementation of a prototype toolkit and its custodian have indicated the general utility of the proposed methods. It is clear that the task of integrating a collection of CAD tools is not a trivial task. Some portions of the task will require highly skilled people (eg. the distribution of existing tools). Other portions of the task are relatively easy but will be time-consuming and tedious (eg. evaluation

of each toolkit drawer and its components). The component evaluation technique and its associated metrics produced absolute and relative measures of the success in achieving a desired set of drawer characteristics. These measures can be useful when considering the internal strengths and weaknesses of a drawer and when selecting the "best" drawer from a collection of similar drawers.

The following chapter concludes this document and presents this author's conclusions and recommendations for future work.

VII. Conclusions and Recommendations

Synopsis

It has been proposed and exemplified that the success of future VLSI design efforts is contingent upon the evolution of a new design philosophy and upon the production of "tall-thin" VLSI design engineers. The lack of an integrated, compatible set of VLSI CAD tools has been identified as a serious problem which prohibits the optimization of the academic programs necessary to produce the required VLSI design engineers.

This thesis proposes that the CAD tool problem may be solved through the development of a VLSI CAD "toolkit". The characteristics of an integrated, technically complete, academically-oriented VLSI CAD toolkit have been presented. A systematic method to integrate CAD tools has been detailed and supported through the development of a set of CAD tool evaluation metrics. Finally, a prototype VLSI CAD toolkit and its custodian were designed, implemented, and evaluated.

Conclusions

It is this author's opinion that all complex entities have evolved from simplistic principles. The development of a complex, intricate structure or idea clearly must begin with the recognition and clear description of its simple basis. This philosophy prevailed throughout this work.

The description of a VLSI CAD toolkit as a simple collection of drawers established a most useful basis for

the detailed toolkit characterization that followed. Desirable toolkit characteristics and properties were assembled from the published comments of VLSI design experts, conversations with AFIT faculty and students, and from this author's limited personal experience with VLSI CAD tools. Admittedly, the toolkit characterization may be biased toward the parochial interests of the author and AFIT. Certainly, the characterization is incomplete since no direct input was obtained from other academic institutions with similar VLSI programs. Although biased and incomplete, the toolkit characterization touches the major needs of VLSI designers as seen in the literature and should provide a viable foundation for future work.

The methodology proposed for the systematic integration of CAD tools also has its basis in simplicity. No new or startling ideas are seen in the proposed integration methods. Rather, the reader will find a logical progression of tasks that form the simple basis of any product evaluation. The complex metrics found so often in software evaluation projects have purposely been avoided. Mathematically naive metrics were presented which were easy to assign, to calculate, and to revise. The proposed metrics appear to have substantial utility as indicators of the "level of presence" of desired toolkit characteristics. The evaluation of the prototype toolkit custodian revealed the strengths and weaknesses of the program that were known to this author a priori. Further application of the metrics by

disinterested parties must be accomplished if the true utility of the metrics is to be determined.

The suggested VLSI CAD toolkit structure evolved in part from this author's exposure to the UNIX computer operating system developed at the Bell Laboratories. The toolkit structure is not dependent upon the presence of that operating system. The toolkit configuration is dependent upon an computer which supports hierarchical directory structures. It is not believed that the toolkit dependence upon hierarchical directory structures is a limitation since the trend in modern computer operating systems seems to favor those structures.

The integration of a collection of VLSI CAD tools into a complete toolkit should be a simple task but it will not be a trivial one. The quantity of work and the spectrum of expertise required will be substantial. The successful implementation of a VLSI CAD toolkit, using the methodology suggested in this work, will enable academic institutions to optimize their VLSI curricula and to produce the types of VLSI design engineers the future demands.

Recommendations

The simple basis of an integrated VLSI CAD toolkit has been provided. The successful evolution of such a toolkit will require substantial future efforts. The tasks described below should be included in those efforts.

1. The productivity of some VLSI courses (and the major thrust of this thesis) should be evaluated through the use of the prototype toolkit and its

associated custodian.

2. The toolkit characterization must be completed. First, all locally interested parties should be provided the opportunity to contribute to the characterization. Second, input to the toolkit characterization should be obtained from all other academic institutions with VLSI curricula.

3. The toolkit metrics proposed herein should be carefully examined. It is recommended that the evaluation of the prototype custodian be accomplished by a group of actual and potential users of VLSI CAD tools. The group should not be limited to those intimately familiar with VLSI design or CAD tools. The intention is to make VSLI design available to many disciplines.

4. The prototype custodian should be rewritten in a language that is generally transportable (eg. "C") and provided to interested parties for their evaluation.

5. The distribution of AFIT CAD tools should be independently accomplished by a group of people who are intimately familiar with the CAD tools.

Bibliography

1. Ayres, Ronald F. VLSI: Silicon Compilation and the Art of Automatic Microchip Design. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1983.

2. Bennett, Jack. "A Database Management System for Design Engineers," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. 268-273. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

3. Bernhard, Robert. "VLSI/LSI Components," IEEE Spectrum, 19 (1): 49-54 (January 1982).

4. Beyls, A.M. and others. "A Design Methodology Based Upon Symbolic Layout and Integrated CAD Tools," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. 872-878. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

5. Boehm, B.W. and others. "Quantitative Evaluation of Software Quality," Models and Metrics for Software Management and Engineering, edited by Victor R. Basili. New York: The Institute of Electrical and Electronics Engineers Inc., 1980.

6. Bracco, A.M. "VLSI In the 80's and 90's, An Industry Perspective," Proceedings of the First Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 21-28. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1982.

7. Breuer, Melvin A. "A Survey of the State-of-the-Art of Design Automation," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

8. Dean, Jeffery S. and Brian P. McCune. Advanced Tools for Software Maintenance. Technical report number RADC-TR-82-313. Contract number F30602-80-C-0176. Rome Air Development Center, Griffiss AFB, New York, December 1982.

9. Drongowski, Paul J. "Teaching The Architectural Design of VLSI Systems" Proceedings of the Second Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 11-17. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1983.

10. Dryer, J.A. "Electronics Design Automation in the 1980's, A View from the Production Side" Proceedings of the First Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 31-35. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1982.

11. Fischetti, Mark A. "VLSI/LSI Components," IEEE Spectrum, 20 (1): 43-47 (January 1983).

12. Fischetti, Mark A. "Solid State." IEEE Spectrum, 21 (1): 58-63 (January 1984).

13. Friedenson, R.A. and others. "Designers Workbench: Delivery of CAD Tools," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. 15-22. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

14. Kingsley, Chris. Earl: An Integrated Circuit Design Language: Technical Report # 5021. Contract number N00014-79-C-0597. Computer Science Department, California Institute of Technology, Pasadena, California, June 1982.

15. Leath, Charles L. and Steven J. Ollanik. "Software Architecture for the Implementation of a Computer-Aided Engineering System," Proceedings of the ACM IEEE 20th Design Automation Conference. 137-142. New York: The Institute of Electrical and Electronics Engineers Inc., 1983.

16. LeBlond, Andre. "CAF: A Computer-Assisted Floorplanning Tool," Proceedings of the ACM IEEE 20th Design Automation Conference. 747-753. New York: The Institute of Electrical and Electronics Engineers Inc., 1983.

17. Losleben, Paul. "Computer Aided Design for VLSI," Very Large Scale Integration (VLSI), Fundamentals and Applications, Volume 5, edited by D.F. Barbe. 89-127. Berlin Heidelberg: Springer-Verlag, 1980.

18. Mayo, John S. "Design Automation - Lessons of the Past, Challenges for the Future," Proceedings of the ACM IEEE 20th Design Automation Conference. 1-2. New York: The Institute of Electrical and Electronics Engineers Inc., 1983.

19. Mayo Robert N. and others. 1983 VLSI Tools: Selected Works by the Original Artists. Computer Science Division, EECS Department, University of California at Berkeley, March 1983.

20. McDonald, J.F. and others. "Use of CAD/CAM in VLSI," Proceedings of the First Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 85-92. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1982.

21. Mead, Carver and Lynn Conway. Introduction to VLSI Systems. Reading, Massachusetts: Addison-Wesley Publishing Company, 1980.

22. Mead, Carver A. "VLSI and Technological Innovations," VLSI 81, Very Large Scale Integration, Proceedings of the first International Conference on Very Large Scale Integration. 3-11. New York: Academic Press, 1981.

23. Meindl, James D. "Microelectronic Circuit Elements," Scientific American, 237 (3): 70-81 (September 1977).

24. Mudge, J. Craig. "VLSI Chip Design at the Crossroads," VLSI 81, Very Large Scale Integration, Proceedings of the first International Conference on Very Large Scale Integration. 205-215. New York: Academic Press, 1981.

25. North, Stephen C. Molding Clay: A Manual for the Clay Layout Language: VLSI Memo #3. Department of Electrical Engineering and Computer Science, Princeton University, Princeton, New Jersey July 1983.

26. Noyce, Robert N. "Microelectronics," Scientific American, 237 (3): 62-69 (September 1977).

27. Oakes, M.F. "The Complete VLSI Design System," VLSI, The Coming revolution in Applications and Design, edited by Rex Rice. 189-197. New York: The Institute of Electrical and Electronics Engineers Inc., 1980.

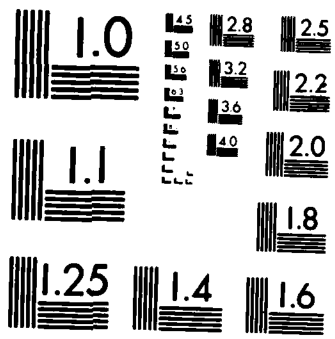
28. Pucacco, Leo R. and others. "Computer Aided Design Integrated in the Electrical Engineering Curriculum," Proceedings of the First Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 51-54. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1982.

29. Rice, Rex. VLSI, The Coming Revolution in Applications and Design. New York: The Institute of Electrical and Electronics Engineers Inc., 1980.

30. Richardson, Fontaine K. "Important Criteria in Selecting Engineering Work Stations," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. 440-444. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

AD-A151 813 THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 2/3

UNCLASSIFIED T R VERMILLION DEC 84 AFIT/GE/ENG/84D-68 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

31. Sapiro, Steve. "Engineering Workstations: Tools or Toys?" Proceedings of the ACM IEEE 20th Design Automation Conference. 79-80. New York: The Institute of Electrical and Electronics Engineers Inc., 1983.

32. Smith, Kent F. and others. "Student-Designed VLSI at the University of Utah," Proceedings of the Second Annual Workshop on Interactive Computing: CAD/CAM: Electrical Engineering Education. 26-30. Silver Spring, Maryland: Institute of Electrical and Electronics Engineers Computer Society Press, 1983.

33. Summey, Larry W. "VHISC: a status report," IEEE Spectrum, 19 (12): 34-39 (December 1982).

34. Szygenda, S.A. "VLSI - A Challenge to CAD," Proceedings of the International Conference on Computer Aided Design and Manufacture of Electronic Components, Circuits, and Systems. 1-6. Institution of Electrical Engineers, London, 1979.

35. "The Solid State Era," Electronics, 53 (9): 215-270, April 1980.

36. Thompson, T. J. "A Utilitarian Approach to CAD," Proceedings of the ACM IEEE Nineteenth Design Automation Conference. 23-29. New York: The Institute of Electrical and Electronics Engineers Inc., 1982.

37. "Thoughts on VLSI Design". Paper distributed in EE 6.95, VLSI Design Laboratory I, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1983.

38. Tomkinson, James H. "UCAD: Building Design Automation with General Purpose Software tools on UNIX," Proceedings of the ACM IEEE 20th Design Automation Conference. 774-787. New York: The Institute of Electrical and Electronics Engineers Inc., 1983.

39. UW/NW VLSI Consortium. VLSI Design Tools Reference Manual. Department of Computer Science, University of Washington, Seattle, Washington, June 1984.

40. vanCleemput, W. M. "The Role of Design Automation in the Design of Digital Systems," Computer-Aided Design Tools for Digital Systems. New York: The Institute of Electrical and Electronics Engineers Inc., 1979.

41. vanCleemput, W.M. "Design Automation Requirements for VLSI," VLSI, The Coming revolution in Applications and Design, edited by Rex Rice. 184-188. New York: The Institute of Electrical and Electronics Engineers Inc., 1980.

42. "VLSI Design Tools". Air Force Institute of Technology, Wright-Patterson AFB, Ohio, January 1983.

Appendix A

High-Level Design of the Toolkit Custodian

Node Index

A1	CUS-0	Provide Toolkit Custodian
A2	CUS0	Provide Toolkit Custodian
A3	CUS1	Assign User Access Level
A4	CUS2	Open Toolkit
A5	CUS21	Open for Authorized User
A6	CUS22	Record Unauthorized User Access Attempt
A7	CUS3	Provide Toolkit
A8	CUS31	Obtain User Input
A9	CUS32	Analyze User Input
A10	CUS33	Respond to User Input
A11	CUS34	Analyze Response
A12	CUS4	Close Toolkit
A13	CUS41	Purge Temporary Storage
A14	CUS42	Update Toolkit Management Data
A15	CUS43	Append Temporary Log to Permanent Log
A16	CUS44	Send Closing Messages

See Reverse Side for SADT Text

CUS-0 Provide Toolkit Custodian

Abstract: This is the environment node. This program implements the VLSI CAD toolkit "custodian". This program provides the interface between the toolkit and toolkit users as well as the interface between the toolkit and its support computer. This program also provides the control mechanism for the toolkit.

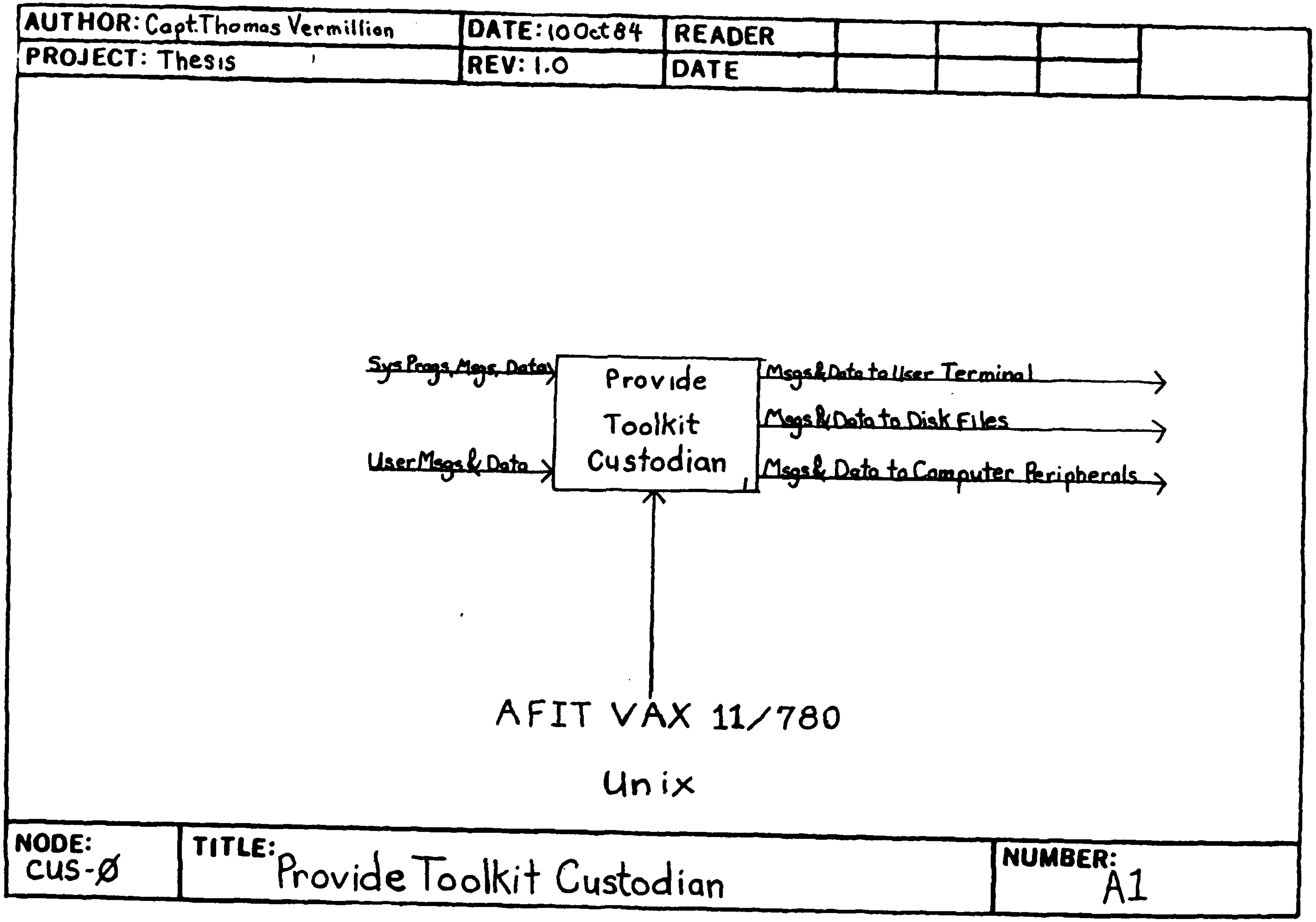


Figure A-1 SADT for Custodian Node CUS-0

See Reverse Side for SADT Text

CUS0 Provide Toolkit Custodian

Abstract: This is the fundamental structure of the VLSI CAD toolkit custodian. The custodian performs four major functions.

CUS1 Assign User Access Level - This process performs a single function; assignment of an access level code to the user requesting access to the VLSI CAD toolkit. The access level will be determined by searching pre-established lists of authorized users for the presence of the requester's user id. In the event the requester's user id appears in more than one list, the access code will reflect an access level which encompasses the access permitted by all matching lists.

CUS2 Open Toolkit - The first activity of this process is the creation of a temporary log in which will be recorded all significant activity during this toolkit session. This process then examines the access level of the user and, if the user is an authorized user, performs the activities necessary to ensure that the system environment, the user interface, and the toolkit components are available and configured to ensure successful execution of all the toolkit components accessible by the user. If the user is not authorized access, the user is counseled concerning the steps necessary to gain access and an entry is made in the toolkit security log concerning this execution attempt.

CUS3 Provide Toolkit - This process forms the core of the custodian. At this level the user requests are obtained and analyzed. If the analysis detects an error in the syntatical or logical structure of a request, then action is taken to resolve the error. Verification of resource availability occurs during the analysis of the user requests. If the user requests successfully pass the analysis step, then the a response to the requests is selected and the appropriate toolkit components are executed. The toolkit response to the user requests in analyzed following completion of the response. Throughout this process appropriate statistics, messages, and data are generated and stored.

CUS4 Close Toolkit - This process "gracefully" closes the toolkit. Storage space which was temporarily reserved for this session is purged. The toolkit management data is updated and stored. The temporary log opened to record this session's activity is appended to the permanent toolkit log. A closing message is generated and the custodian ceases execution.

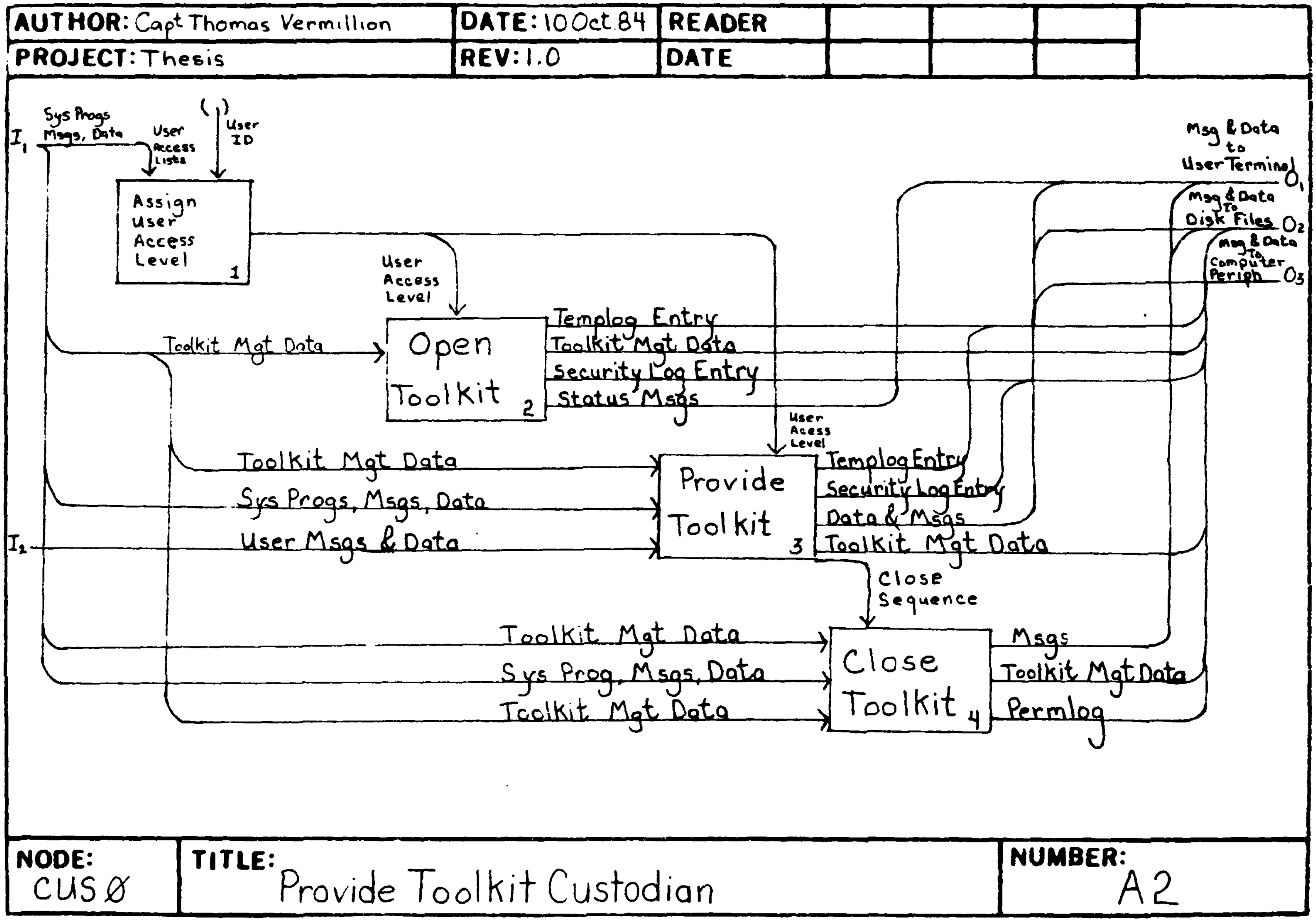


Figure A-2 SADT for Custodian Node CUS0

See Reverse Side for SADT Text

CUS1 Assign User Access Level

Abstract: This level performs a single function; assignment of an access level code to the user requesting access to the VLSI CAD toolkit. The access level will be determined by searching pre-established lists of authorized users for the presence of the requester's user id. In the event the requester's user id appears in more than one list, the access code will reflect an access level which encompasses the access permitted by all matching lists.

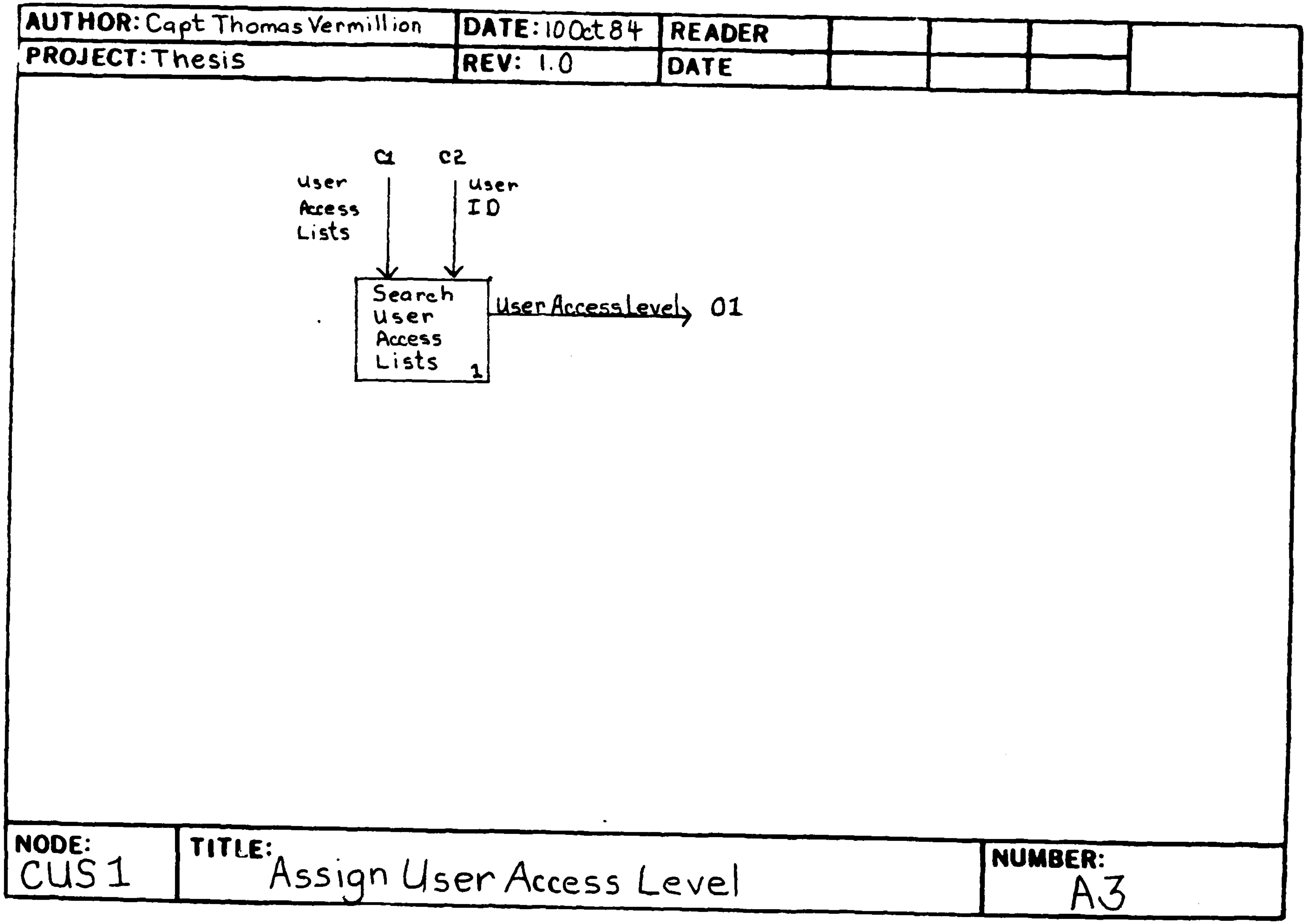


Figure A-3 SADT for Custodian Node CUS1

See Reverse Side for SADT Text

CUS2 Open Toolkit

Abstract: The first activity of this process is the creation of a temporary log in which will be recorded all significant activity during this toolkit session. This process then examines the access level of the user and, if the user is an authorized user, performs the activities necessary to ensure that the system environment, the user interface, and the toolkit components are available and configured to ensure successful execution of all the toolkit components accessible by the user. If the user is not authorized access, the user is counseled concerning the steps necessary to gain access and an entry is made in the toolkit security log concerning this execution attempt.

CUS21 Open for Authorized User - This process performs the necessary activities to open the toolkit for use by an authorized toolkit user. Those components of the toolkit management data which are affected by an authorized toolkit access are updated and appropriate entries are made in the temporary log. Status messages are provided the user to ensure the required feedback is provided.

CUS22 Record Unauthorized User Access Attempt - This process records an attempt to open the toolkit by an unauthorized user. The user is provided instructions concerning the steps necessary to gain access authorization.

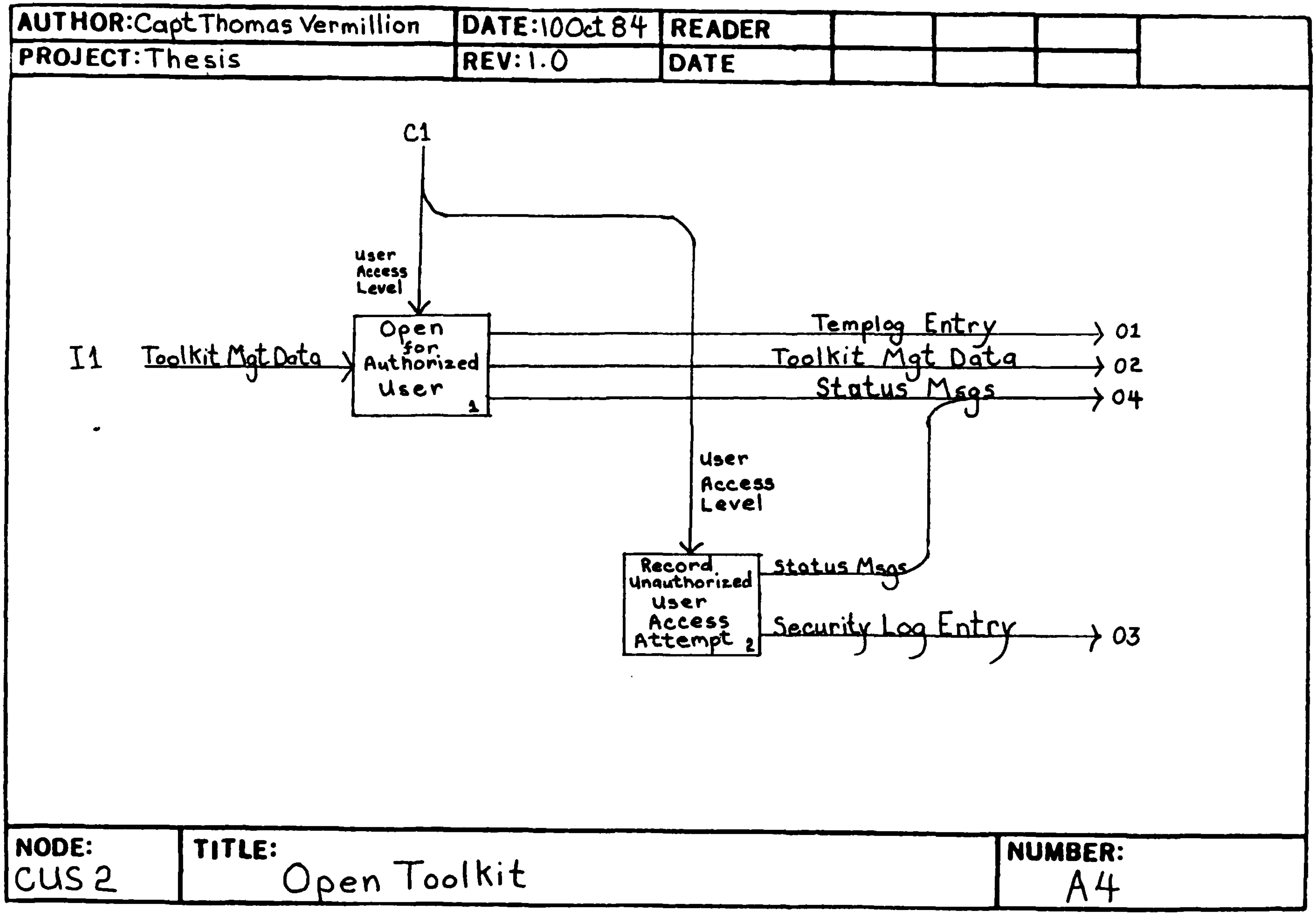


Figure A-4 SADT for Custodian Node CUS2

See Reverse Side for SADT Text

CUS3 Provide Toolkit

Abstract: This process forms the core of the custodian. At this level the user requests are obtained and analyzed. If the analysis detects an error in the syntatical or logical structure of a request, then action is taken to resolve the error. Verification of resource availability occurs during the analysis of the user requests. If the user requests successfully pass the analysis step, then the a response to the requests is selected and the appropriate toolkit components are executed. The toolkit response to the user requests in analyzed following completion of the response. Throughout this process appropriate statistics, messages, and data are generated and stored.

CUS31 Obtain User Input - This process performs the activities necessary to ascertain the skill level of the user, present selection options to the user, and actually obtain the user's input. The selection options provided to the user are based upon an analysis of the last toolkit response to a user request as well as the current status of the toolkit's operating environment. Appropriate log entries are stored. Attempts to violate the security of the toolkit are detected and recorded. Help is provided as necessary.

CUS32 Analyze User Input - *This process parses the user input and analyzes* the input for syntatical and logical errors. Attempts to violate toolkit security are detected and recorded. An action code is generated which corresponds to the user's component selection and statistics are generated concerning the user selection and the parameters selected by the user for use during the execution of the selected component. Help is provided as necessary.

CUS33 Respond to User Input - This process executes the toolkit components necessary to effect a complete response to the user input. This process manages and controls the execution sequence and data flow until the response is completed. Continuous feedback is provided the user during component execution. A response code is generated which reflects whether or not the response was successfully completed. If an error is detected during a response, an attempt is made to recover from the error. Help is provided as necessary.

CUS34 Analyze Response - The process performs the activities necessary to examine and analyze the latest toolkit response to a user request. If the response was terminated due to an error, detailed information concerning the error is provided to the user. Help is provided to the user to aid in the interpretation and correction of the error. An analysis code is generated which reflects the overall successfulness of the response.

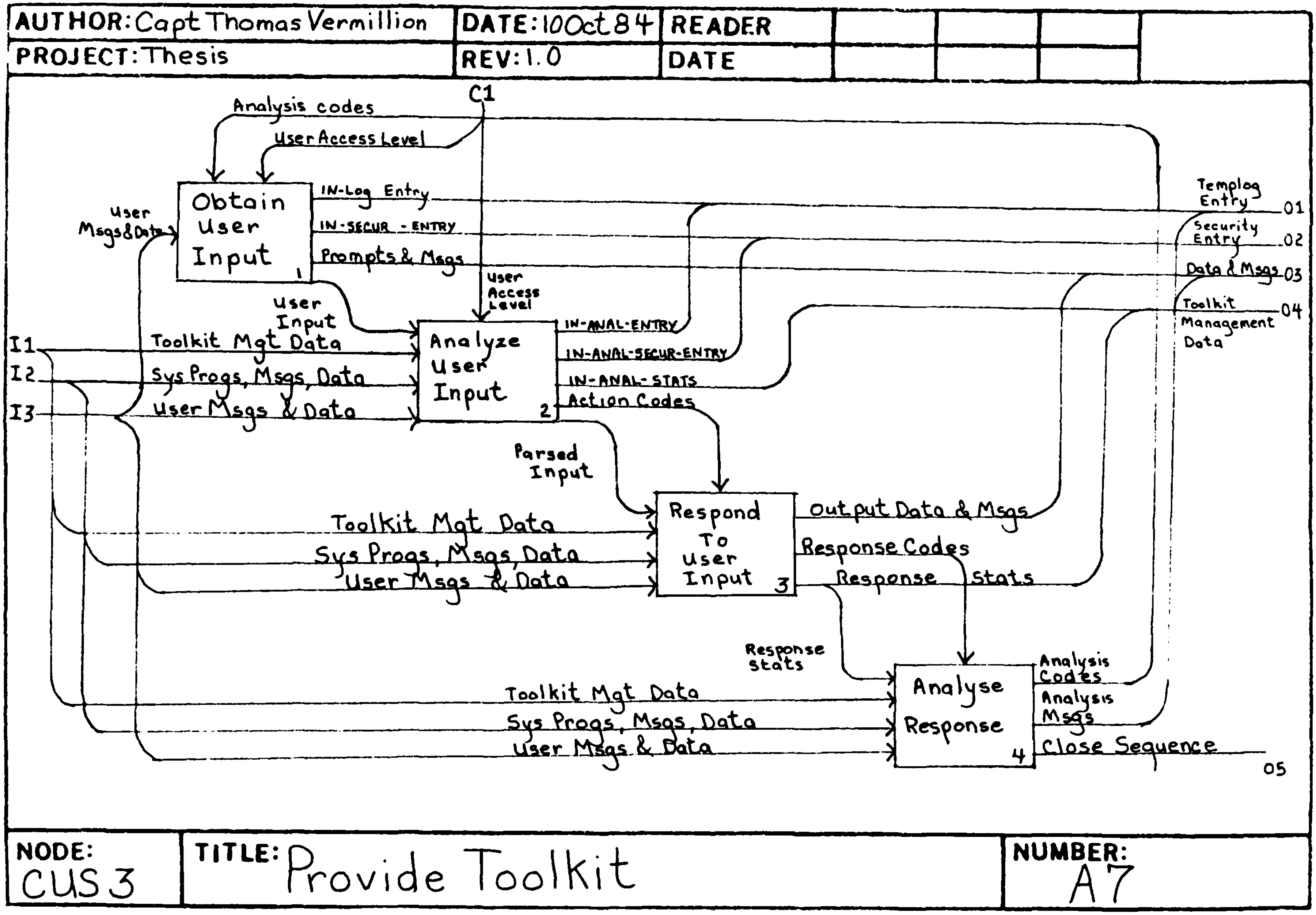


Figure A-5 SADT for Custodian Node CUS3

See Reverse Side for SADT Text

CUS4 Close Toolkit

Abstract: This process "gracefully" closes the toolkit. Storage space which was temporarily reserved for this session is purged. The toolkit management data is updated and stored. The temporary log opened to record this session's activity is appended to the permanent toolkit log. A closing message is generated and the custodian ceases execution.

CUS41 Purge Temporary Storage - This process purges the toolkit temporary storage are of all files generated during this session which are no longer required. Some files may remain if the user has requested that they be added to the permanent storage area. If files remain for addition to the permanent storage area, then an appropriate storage request is generated. This message will be presented to those responsible for the toolkit configuration and maintenance.

CUS42 Update Toolkit Management Data - This process performs the activities necessary to bring all of the toolkit management data up-to-date with respect to the activities that occurred during this user session. The toolkit management data includes requests to move temporary storage files to the permanent storage area. Session statistics are provided to the user.

CUS43 Append Temporary Log to Permanent Log - This process appends the temporary log generated and maintained during this session to the current permanent toolkit log. The temporary log is then removed and discarded.

CUS44 Send Closing Messages - This process generates appropriate messages to let the user know that the session has "gracefully" ended. Messages are also sent to the toolkit maintainers concerning matters which have arisen during this session which require attention (eg. a request to make an addition to the toolkit permanent storage or user comments).

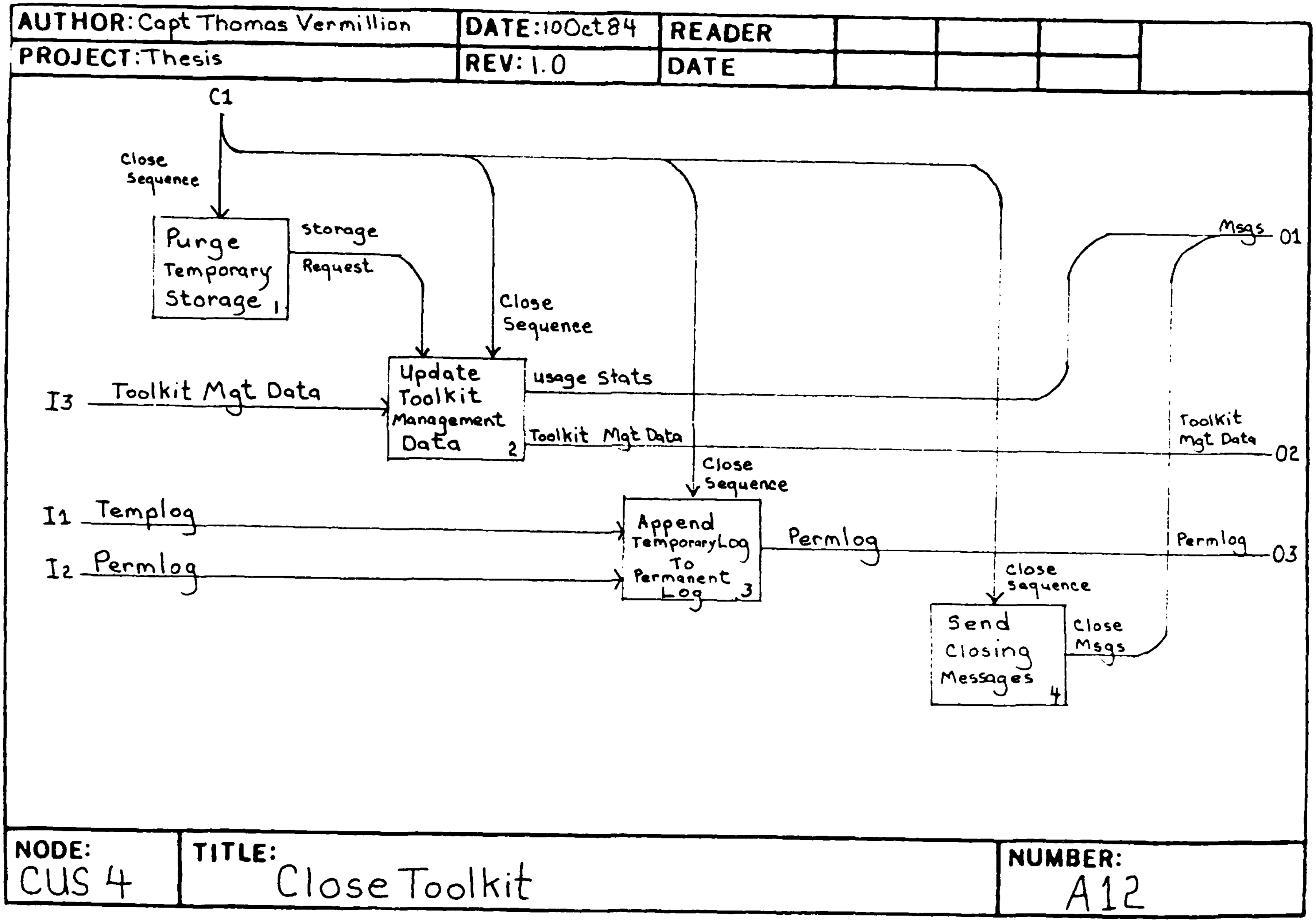


Figure A-6 SADT for Custodian Node CUS4

Data Dictionary

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS-0
NAME: Provide Toolkit Custodian
INPUTS: Sys Progs,Msgs, Data
User Msgs & Data
OUTPUTS: Msgs & Data to User Terminal
Msgs & Data to Disk Files
Msgs & Data to Computer Peripherals
CONTROLS: None.
MECHANISMS: AFIT VAX 11/780 with Unix Os.
DESCRIPTION: Environment Node for the VLSI CAD Toolkit Custodian. This custodian provides the interface between the toolkit and the outside world. The custodian also provides complete control of the toolkit execution and dataflow.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS1
NAME: Assign User Access Level
INPUTS: None.
OUTPUTS: User Access level
CONTROLS: User Access Lists
User Id
MECHANISMS: None.
DESCRIPTION: This process performs a single function; assignment of an access level code to the user requesting access to the VLSI CAD toolkit. The access level will be determined by searching pre-established lists of authorized users for the presence of the requester's user id. In the event the requester's user id appears in more than one list, the access code will reflect an access level which encompasses the access permitted by all matching lists.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS2
NAME: Open Toolkit
INPUTS: Toolkit Mgt Data
OUTPUTS: Templog Entry
Toolkit Mgt Data
Status Msgs
CONTROLS: User Access Level
MECHANISMS: None.
DESCRIPTION: The first activity of this process is the creation of a temporary log in which will be recorded all significant activity during this toolkit session. This process then examines the access level of the user and, if the user is an authorized user, performs the activities necessary to ensure that the system environment, the user interface, and the toolkit components are available and configured to ensure successful execution of all the toolkit components accessible by the user. If the user is not authorized access, the user is counseled concerning the steps necessary to gain access and an entry is made in the toolkit security log concerning this execution attempt.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS21
NAME: Open for Authorized User
INPUTS: Toolkit Mgt Data
OUTPUTS: Templog Entry
Toolkit Mgt Data
Status Msgs
CONTROLS: User Access Level
MECHANISMS: None.
DESCRIPTION: This process performs the necessary activities to open the toolkit for use by an authorized toolkit user. Those components of the toolkit management data which are affected by an authorized toolkit access are updated and appropriate entries are made in the temporary log. Status messages are provided the user to ensure the required feedback is provided.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS22
NAME: Record Unauthorized User Access Attempt
INPUTS: None.
OUTPUTS: Status Msgs
Security Log Entry
CONTROLS: User Access Level
MECHANISMS: None.
DESCRIPTION: This process records an attempt to open the toolkit by an unauthorized user. The user is provided instructions concerning the steps necessary to gain access authorization.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS3
NAME: Provide Toolkit
INPUTS: Toolkit Mgt Data
Sys Progs, Msgs, Data
User Msgs & Data
OUTPUTS: Templog Entry
Security Log Entry
Data & Msgs
Toolkit Mgt Data
CONTROLS: User Access Level
MECHANISMS: None.
DESCRIPTION: This process forms the core of the custodian. At this level the user requests are obtained and analyzed. If the analysis detects an error in the syntatical or logical structure of a request, then action is taken to resolve the error. Verification of resource availability occurs during the analysis of the user requests. If the user requests successfully pass the analysis step, then the a response to the requests is selected and the appropriate toolkit components are executed. The toolkit response to the user requests in analyzed following completion of the response. Throughout this process appropriate statistics, messages, and data are generated and stored.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS31
NAME: Obtain User Input
INPUTS: User Msgs & Data
OUTPUTS: In_Log_Entry
In_Secur_Entry
Prompts & Msgs
CONTROLS: Analysis Code
User Access Level
MECHANISMS: None.
DESCRIPTION: This process performs the activities necessary to ascertain the skill level of the user, present selection options to the user, and actually obtain the user's input. The selection options provided to the user are based upon an analysis of the last toolkit response to a user request as well as the current status of the toolkit's operating environment. Appropriate log entries are stored. Attempts to violate the security of the toolkit are detected and recorded. Help is provided as necessary.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS32
NAME: Analyze User Input
INPUTS: Toolkit Mgt Data
Sys Progs, Msgs, Data
User Msgs & Data
User Input
OUTPUTS: In_Anal_Entry
In_Anal_Sec_Entry
In_Anal_Stats
Action_Code
CONTROLS: User Access Level
MECHANISMS: None.
DESCRIPTION: This process parses the user input and analyzes the input for syntatical and logical errors. Attempts to violate toolkit security are detected and recorded. An action code is generated which corresponds to the user's component selection and statistics are generated concerning the user selection and the parameters selected by the user for use during the execution of the selected component. Help is provided as necessary.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS33
NAME: Respond To User Input
INPUTS: Toolkit Mgt Data
Sys Progs, Msgs, Data
User Msgs & Data
Parsed Input
OUTPUTS: Output Data & Msgs
Response Code
Response Stats
CONTROLS: Action Code
MECHANISMS: None.
DESCRIPTION: This process executes the toolkit components necessary to effect a complete response to the user input. This process manages and controls the execution sequence and data flow until the response is completed. Continuous feedback is provided the user during component execution. A response code is generated which reflects whether or not the response was successfully completed. If an error is detected during a response, an attempt is made to recover from the error. Help is provided as necessary.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS34
NAME: Analyze Response
INPUTS: Toolkit Mgt Data
Sys Progs, Msgs, Data
User Msgs & Data
Response Stats
OUTPUTS: Analysis Code
Analysis Msgs
Close Sequense
CONTROLS: Response Code
MECHANISMS: None.
DESCRIPTION: The process performs the activities necessary to examine and analyze the latest toolkit response to a user request. If the response was terminated due to an error, detailed information concerning the error is provided to the user. Help is provided to the user to aid in the interpretation and correction of the error. An analysis code is generated which reflects the overall successfulness of the response.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS4
NAME: Close Toolkit
INPUTS: Templog
Permlog
Toolkit Mgt Data
OUTPUTS: Msgs
Toolkit Mgt Data
Permlog
CONTROLS: Close Sequence
MECHANISMS: None.
DESCRIPTION: This process "gracefully" closes the toolkit. Storage space which was temporarily reserved for this session is purged. The toolkit management data is updated and stored. The temporary log opened to record this session's activity is appended to the permanent toolkit log. A closing message is generated and the custodian ceases execution.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS41
NAME: Purge Temporary Storage
INPUTS: None.
OUTPUTS: Storage Request
CONTROLS: Close Sequence
MECHANISMS: None.
DESCRIPTION: This process purges the toolkit temporary storage are of all files generated during this session which are no longer required. Some files may remain if the user has requested that they be added to the permanent storage area. If files remain for addition to the permanent storage area, then an appropriate storage request is generated. This message will be presented to those responsible for the toolkit configuration and maintenance.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS42
NAME: Update Toolkit Management Data
INPUTS: Toolkit Mgt Data
OUTPUTS: Usage Stats
Toolkit Mgt Data
CONTROLS: Storage Request
Close Sequence
MECHANISMS: None.
DESCRIPTION: This process performs the activities necessary to bring all of the toolkit management data up-to-date with respect to the activities that occurred during this user session. The toolkit management data includes requests to move temporary storage files to the permanent storage area. Session statistics are provided to the user.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS43
NAME: Append Temporary Log to Permanent Log
INPUTS: Templog
Permlog
OUTPUTS: Permlog
CONTROLS: Close Sequence
MECHANISMS: None.
DESCRIPTION: This process appends the temporary log generated and maintained during this session to the current permanent toolkit log. The temporary log is then removed and discarded.
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: ACTIVITY
DATE: 10 October 1984
NUMBER: CUS44
NAME: Send Closing Messages
INPUTS: None.
OUTPUTS: Close Msgs
CONTROLS: Close Sequence
MECHANISMS: None.
DESCRIPTION: This process generates appropriate messages to let the user know that the session has "gracefully" ended. Messages are also sent to the toolkit maintainers concerning matters which have arisen during this session which require attention (eg. a request to make an addition to the toolkit permanent storage or user comments).
RELATED REQUIREMENT NUMBER: N/A
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Sys Progs, Msgs, Data
DESCRIPTION: Executable programs, data, and messages which are integral parts of the support computer operating system.
SOURCES: Support Computer Operating System
DESTINATIONS: CUS0 CUS1 CUS3
COMPOSITION: Executable Programs, Data generated by the executable programs, data stored in system-level disk files or system-level operating system variables, and messages generated by the executable programs or by the operating system.
DATA CHARACTERISTICS: Executable Programs are binary. All other data is composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: User Msgs & Data
DESCRIPTION: Data and messages, directly accessible to the user, which are received from the user terminal or whose location is identified by the user.
SOURCES: User Terminal
DESTINATIONS: CUS0 CUS3
COMPOSITION: Data generated by user-accessible executable programs, stored in user-accessible disk files or user-owned, system-level variables, and messages generated by the executable programs or manually by the user.
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Msgs & Data To User Terminal
DESCRIPTION: Messages and data sent directly to the user's terminal.
SOURCES: CUSO
DESTINATIONS: User Terminal
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Msgs & Data to Disk Files
DESCRIPTION: Messages and data sent to disk files
SOURCES: CUSO
DESTINATIONS: Disk Files
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Msgs & Data to Computer Peripherals
DESCRIPTION: Messages and data sent to the peripherals attached to the support computer system or to the user terminal.
SOURCES: CUSO
DESTINATIONS: Peripherals attached to the support computer system or to the user terminal.
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Msgs & Data to Computer Peripherals
DESCRIPTION: Messages and data sent to the peripherals attached to the support computer system or to the user terminal.
SOURCES: CUS0
DESTINATIONS: Peripherals attached to the support computer system or to the user terminal.
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: User Access Lists
DESCRIPTION: Lists of unique user identifiers (user ids) which are segregated according to the various levels of toolkit access permitted the users.
SOURCES: Resident in the support computer disk storage area.
DESTINATIONS: CUS1
COMPOSITION: Unique user identifiers
PART OF: Sys Progs, Msgs, Data
DATA CHARACTERISTICS: ASCII characters
VALUES: Range of values permitted by the host computer operating system for the unique identification of users.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: User Id
DESCRIPTION: The unique identifier assigned by the operating system for the current user. Usually stored in an operating system variable.
SOURCES: Support Computer Operating System.
DESTINATIONS: CUS1
COMPOSITION: Unique user identifier
DATA CHARACTERISTICS: ASCII characters
VALUES: Range of values permitted by the host computer operating system for the unique identification of users.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: User Access Level
DESCRIPTION: Descriptor which indicates the level of access that is to be given the current toolkit user. This level of access is used to determine: 1) which toolkit components may be executed by the user, 2) the types and quantities of toolkit data that will be accessible to the user, and 3) the priority of this user's requests relative to other toolkit processing tasks (including other user requests).
SOURCES: CUS1
DESTINATIONS: CUS2 CUS21 CUS22 CUS3 CUS31 CUS32
COMPOSITION: A number is used to represent the level of access.
DATA CHARACTERISTICS: A single integer.
VALUES: 0 : No access permitted - TBD based upon highest level of access.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Templog Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit "event". This entry is stored in the temporary log assigned to the current user session.
SOURCES: CUS2 CUS21 CUS3
DESTINATIONS: Toolkit temporary log
COMPOSITION: Message describing the occurrence of a toolkit "event".
PART OF: Msgs & Data to Disk Files
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Toolkit Mgt Data
DESCRIPTION: Data necessary to manage toolkit component execution, the toolkit configuration, and all toolkit resources.
SOURCES: Resides in support computer system disk files and variables. Is updated/modified by CUS2 CUS3 CUS4
DESTINATIONS: CUS2 CUS3 CUS4
COMPOSITION: Toolkit access data, user request data, toolkit resource data, operating system resource data, toolkit component performance data, and toolkit maintenance data and attention flags.
PART OF: Msgs & Data to Disk Files
DATA CHARACTERISTICS: All data and attention flags are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Security Log Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit security "event". This entry is stored in the toolkit security log.
SOURCES: CUS2 CUS3
DESTINATIONS: Toolkit security log
COMPOSITION: Message describing the occurrence of a toolkit security "event".
PART OF: Msgs & Data to Disk Files
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Status Msgs
DESCRIPTION: Messages generated that provide the user with feedback concerning the tasks being processed by the toolkit custodian when the toolkit is being opened for use.
SOURCES: CUS2 CUS21 CUS22
DESTINATIONS: User Terminal
COMPOSITION: Message describing the current task being processed by the toolkit custodian.
PART OF: Msgs & Data to User Terminal
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Data & Msgs
DESCRIPTION: Data and messages generated by the toolkit when performing the tasks necessary to process a user's request.
SOURCES: CUS3
DESTINATIONS: User terminal
Disk files
Computer peripherals attached to the support computer or to the user terminal.
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
PART OF: Msgs & Data to User Terminal
Msgs & Data to Disk Files
Msgs & Data to Computer Peripherals
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Close Sequence
DESCRIPTION: An ordered sequence of codes which identify the tasks which are to be accomplished by the toolkit custodian when closing the toolkit.
SOURCES: CUS3 CUS34
DESTINATIONS: CUS4 CUS41 CUS42 CUS43 CUS44
COMPOSITION: An ordered sequence of task codes.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
VALUES: TBD - Task codes to be assigned during toolkit construction.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Templog
DESCRIPTION: A diskfile containing entries describing toolkit "events". A toolkit event is any user entry, toolkit output message, or toolkit activity which would play an essential part in any reconstruction of the user session. The entries are maintained in the order-of-arrival and may be date/time stamped. Used to maintain the "history" of a particular user toolkit session.
SOURCES: Resides in the temporary storage area of the toolkit. One log is created for each user session.
DESTINATIONS: CUS4 CUS43
COMPOSITION: In_Log_Entry
Templog Entry
In_Anal_Entry
Analysis Msgs
PART OF: Msgs & Data to Disk Files
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: Temporary Log

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Permlog
DESCRIPTION: A diskfile containing entries describing toolkit events. The entries are maintained in the order-of-arrival and may be date/time stamped. Used to maintain the "history" of the toolkit for a particular period of interest.
SOURCES: Resides in the permanent storage area of the toolkit. One log is maintained for each toolkit historical segment (eg. by month or by quarter or by year).
DESTINATIONS: CUS4 CUS43
COMPOSITION: Concattenated "temporary logs" (Templog):
In_Log_Entry
Templog Entry
In_Anal_Entry
Analysis Msgs
PART OF: Msgs & Data to Disk Files
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: Permanent Log

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Msgs
DESCRIPTION: Messages generated during the closing of the toolkit that provide the user with feedback concerning the status of the toolkit and their toolkit session.
SOURCES: CUS4
DESTINATIONS: User Terminal
COMPOSITION: Message describing the status of the toolkit and the current toolkit session.
PART OF: Msgs & Data to User Terminal
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: In_Log_Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit "event" during acquisition of the user input and messages. This entry is stored in the temporary log assigned to the current user session.
SOURCES: CUS31
DESTINATIONS: Toolkit temporary log
COMPOSITION: Message describing the occurrence of a toolkit "event" during acquisition of user input data and messages.
PART OF: Templog Entry
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: In_Secur_Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit security "event" during acquisition of the user input and messages. This entry is stored in the temporary log assigned to the current user session.
SOURCES: CUS31
DESTINATIONS: Toolkit security log
COMPOSITION: Message describing the occurrence of a toolkit security "event" during acquisition of user input data and messages.
PART OF: Security Log Entry
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Prompts & Msgs
DESCRIPTION: Prompts and messages generated during the acquisition of user input messages and data that provide the user with: 1) selection options, 2) feedback concerning input selections, 3) Input assistance, and 4) error responses.
SOURCES: CUS11
DESTINATIONS: User Terminal
COMPOSITION: Messages and data which originate from: 1) the custodian, 2) toolkit variables, 3) toolkit help files, and 4) the user (echoed from user input).
PART OF: Data & Msgs
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Analysis Codes
DESCRIPTION: Codes which identifies the level of success achieved by the toolkit during a response to a user request. As a minimum, the codes must identify whether or not the response was successful or failed. Ideally, the codes will indicate the portions of the user request that were successfully completed and those which were not completed. If portions of the user request were not successfully completed, then the codes should identify the reasons for failure.
SOURCES: CUS34
DESTINATIONS: CUS31
COMPOSITION: Codes identifying the success or failure of each toolkit response to a user request.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
VALUES: TBD - Analysis codes to be assigned during toolkit construction.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: User Input
DESCRIPTION: The exact input acquired from the user terminal.
SOURCES: CUS31
DESTINATIONS: CUS32
COMPOSITION: The exact input acquired from the user terminal.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: In_Anal_Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit "event" during analysis of the user input and messages. This entry is stored in the temporary log assigned to the current user session.
SOURCES: CUS32
DESTINATIONS: Toolkit temporary log
COMPOSITION: Message describing the occurrence of a toolkit "event" during analysis of user input data and messages.
PART OF: Templog Entry
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: In_Anal_Sec_Entry
DESCRIPTION: Message generated by the toolkit custodian which indicates the occurrence of a toolkit security "event" during analysis of the user input and messages. This entry is stored in the temporary log assigned to the current user session.
SOURCES: CUS33
DESTINATIONS: Toolkit security log
COMPOSITION: Message describing the occurrence of a toolkit security "event" during analysis of user input data and messages.
PART OF: Security Log Entry
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: In_Anal_Stats
DESCRIPTION: These are statistics developed during the analysis of a particular user input.
SOURCES: CUS32
COMPOSITION: Statistics concerning: 1) individual user component selections, 2) errors detected in the user input, 3) status of support computer resources at the time of the user input request, and 4) status of the toolkit resources at the time of the user input request.
PART OF: Toolkit Mgt Data
DATA CHARACTERISTICS: All statistics are composed of binary representations of characters (ASCII) or numbers. Ideally, the statistics will be represented as a collection of unique names with appropriate codes or frequencies attached to each name as the result of this analysis.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Action Codes
DESCRIPTION: An ordered sequence of codes which identify the responses which are to be accomplished by the toolkit custodian when responding to a user request.
SOURCES: CUS32
DESTINATIONS: CUS33
COMPOSITION: An ordered sequence of response codes.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
VALUES: TBD - Response codes to be assigned during toolkit construction.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Parsed Input
DESCRIPTION: A collection of "words" extracted from the user input. The "words" are selected in such a manner as to enable the construction of a response to the user request. For example, the user input will be parsed into words that identify: 1) identification of the input data (files) necessary to process the user request, 2) identification of the output data (files) necessary to store data and messages resulting from the request, and 3) parameters necessary to configure toolkit components in a manner to ensure user-selected component options are implemented.
SOURCES: CUS32
DESTINATIONS: CUS33
COMPOSITION: A collection of "words" and phrases extracted from the user input.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Output Data & Msgs
DESCRIPTION: Data and messages generated by the toolkit when performing the tasks necessary to respond to a user's request.
SOURCES: CUS33
DESTINATIONS: User terminal
Disk files
Computer peripherals attached to the support computer or to the user terminal.
COMPOSITION: Data generated by toolkit components, stored in toolkit-accessible disk files or in toolkit or system-level variables, and messages generated by toolkit components.
PART OF: Data & Msgs
DATA CHARACTERISTICS: All data and messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Response Codes
DESCRIPTION: Codes which identifies the completion status of each task generated during a response to a user request. If a task or tasks were not successfully completed, then the code should identify the individual reasons for failure.
SOURCES: CUS33
DESTINATIONS: CUS34
COMPOSITION: Codes identifying the success or failure of each toolkit task generated in response to a user request.
DATA CHARACTERISTICS: Binary representations of characters (ASCII) or numbers.
VALUES: TBD - Response codes to be assigned during toolkit construction.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Response Stats
DESCRIPTION: These are statistics developed during the toolkit response to a user input.
SOURCES: CUS33
DESTINATIONS: CUS34
COMPOSITION: Statistics concerning: 1) individual component execution times, 2) errors detected during component execution, 3) status of support computer resources at the beginning and end of each component execution and 4) status of the toolkit resources at the beginning and end of each component execution.
PART OF: Toolkit Mgt Data
DATA CHARACTERISTICS: All statistics are composed of binary representations of characters (ASCII) or numbers. Ideally, the statistics will be represented as a collection of unique names with appropriate codes or frequencies attached to each name as the result of the current response.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Analysis Msgs
DESCRIPTION: Messages generated during the analysis of the toolkit response to the last user request. These messages provide the user with feedback concerning toolkit responses during their toolkit session and also act as input entries into the session "history".
SOURCES: CUS34
DESTINATIONS: User Terminal   Disk Files
COMPOSITION: Message describing the analysis of a toolkit response to a user request.
PART OF: Data & Msgs
Templog Entry
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Storage Request
DESCRIPTION: A three-component indicator that the user wishes to have data currently residing in temporary storage moved to the toolkit permanent storage area.
SOURCES: CUS41
DESTINATIONS: CUS42
COMPOSITION: Contains three parts: The date/time of the request, the unique user id of the user requesting the storage transfer, and the name of the data that is to be moved. The request is contained on a single line with each component separated by a single space.
DATA CHARACTERISTICS: All requests are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Usage Stats
DESCRIPTION: These are statistics developed during a a particular user session which are provided to the user at the end of the session. The intention is to provide the user with a "snapshot" of the session and the resources required during the session.
SOURCES: CUS42
COMPOSITION: Although the actual collection of statistics may vary at the request of the user, as a minimum the following statistics will be provided: 1) component selection frequencies, 2) The total number of errors detected during user inputs, 3) status of support computer resources at the beginning and end of the toolkit session and 4) status of the toolkit resources at the beginning and end of the toolkit session.
PART OF: Msgs
DATA CHARACTERISTICS: All statistics are composed of binary representations of characters (ASCII) or numbers. Ideally, the statistics will be represented as a collection of unique names with appropriate codes or frequencies attached to each name as the result of this analysis.
ALIASES: None.

TYPE: DATA ELEMENT
DATE: 10 October 1984
NAME: Close Messages
DESCRIPTION: Messages generated during the closing of the toolkit. These are messages sent to the user reporting that the toolkit has been successfully closed and that control is being returned to the support computer operating system.
SOURCES: CUS44
DESTINATIONS: User terminal
COMPOSITION: Messages generated by the custodian.
PART OF: Msgs
DATA CHARACTERISTICS: All messages are composed of binary representations of characters (ASCII) or numbers.
ALIASES: None.

Appendix B: Structure of A Prototype AFIT VLSI CAD Toolkit

Prototype Implementation of ToolKit Drawer Structure Using UNIX Hierarchical Directories

A prototype implementation of the VLSI CAD toolkit drawer structure has been accomplished on the AFIT VAX 11/780 computer. This AFIT computer system employs the UNIX operating system which permits the construction of hierarchical disk directories. The directory feature of UNIX was exploited to create a toolkit whose drawers are actually UNIX directories. The drawer-to-UNIX directory mappings for each toolkit compartment are shown below. The directory hierarchy is presented in the next section.

TOOLKIT COMPONENT NAME	UNIX DIRECTORY NAME
VLSI CAD TOOLKIT	vlsi_tool_kit
MANAGEMENT TOOLS DRAWERS	mgt_tools
Configuration	configuration
Data Translation	data_transl
Resource Management	resource_mgt
Data Analysis	data_analysis
Documentation Management	doc_mgt
Environmental Control	environ_ctrl
Data Archival	data_archival
Data Access Control	data_access
Data Security	data_security
Data Storage Management	data_sto_mgt
Report Generation	report_gen

MANAGEMENT TOOLS (Contd.)

Standards Management	stands_mgt
Library Management	lib_mgt
Schedule Management	sched_mgt

TOOLKIT COMPONENT NAME	UNIX DIRECTORY NAME
DESIGN TOOLS DRAWERS	design_tools
Architectural	architecture
Logic	logic
Floorplan	floorplan
Layout	layout
Interconnection	connection
Topological Analysis	topoanalysis
Extraction	extraction
Timing Analysis	timing_analy
Electrical Analysis	elect_analy
Mask Data Generation	mask_data_gen
Design Verification	design_verify
Design Presentation	design_present
Utilities	utilities
INFORMATION STORAGE DRAWERS	info_storage
TEMPORARY STORAGE DRAWER	temp_storage
Documentation	temp_doc
Libraries	temp_libs
Statistics	temp_stats

TEMPORARY STORAGE (Contd.)	
Schedules	temp_sched
Design Data	temp_data
Source Code	temp_sources
Aux Data	temp_aux_data

TOOLKIT COMPONENT NAME	UNIX DIRECTORY NAME
PERMANENT STORAGE DRAWER	perm_storage
Documentation	perm_doc
Libraries	perm_libs
Statistics	perm_stats
Schedules	perm_sched
Design Data	perm_data
Source Code	perm_sources
Aux Data	perm_aux_data

Hierarchical Toolkit Directory Structure

The hierarchical srtucture of the of VLSI CAD toolkit is shown in the diagram that follows. The lines connecting directories or groups of directories represent a change in level to that directory (group). Directories listed without intervening lines are at the same level.

The highest level directory is the "vlsi_tool_kit" level. The toolit custodian should "reside" at the highest level directory in the toolkit structure.

A prototype directory structure is represented by the diagram that follows.

```
                         vlsi_tool_kit
                               |
   ---------------------------------------------------
   |                           |                      |
mgt_tools                 info_storage           design_tools
   |                           |                      |
   |                    ---------------               |
   |                    |             |               |
   |               temp_storage   perm_storage        |
   |                    |             |               |
configuration      temp_doc       perm_doc       architecture
data_transl        temp_libs      perm_libs      logic
resource_mgt       temp_stats     perm_stats     floorplan
data_analysis      temp_sched     perm_sched     layout
doc_mgt            temp_data      perm_data      connection
environ_ctrl       temp_sources   perm_sources   topoanalysis
data_archival      temp_aux_data  perm_aux_data  extraction
data_access                                      timing_analy
data_security                                    elect_analy
data_sto_mgt                                     mask_data_gen
report_gen                                       design_verify
stands_mgt                                       design_present
lib_mgt                                          utilities
sched_mgt
```

The UNIX "C-Shell" Script Used to Build The Toolkit Hierarchical Directory Structure

```
# UNIX C-SHELL Script to build hierarchical directories for the
# VLSI CAD TOOL KIT (drawers)
#
# Written by:  Capt Thomas R. Vermillion
#              AFIT/EN   : GE84D
#              1 October 1984
#
#
echo ' '
echo 'VLSI CAD TOOLKIT DIRECTORY CONSTRUCTION SCRIPT  V 1.0'
echo ' '
echo -n 'Creating Main Toolkit Directory ... '
# main tool kit directory - custodian lives here
mkdir vlsi_tool_kit
chmod go+rxw vlsi_tool_kit
echo 'DONE'
echo  -n 'Creating Major Drawer Directories ... '
#
cd vlsi_tool_kit
# major drawer sections
#
mkdir mgt_tools
mkdir info_storage
mkdir design_tools
chmod go+rwx *
echo 'DONE'
echo -n 'Creating Individual Management Tool Directories. '
#
#management tools drawers
cd mgt_tools
#individual management tools
#
mkdir configuration
mkdir data_transl
mkdir resource_mgt
mkdir data_analysis
mkdir doc_mgt
mkdir environ_ctrl
mkdir data_archival
mkdir data_access
mkdir data_security
mkdir data_stor_mgt
mkdir report_gen
mkdir stands_mgt
mkdir lib_mgt
mkdir sched_mgt
chmod go+rwx *
echo 'DONE'
echo  -n 'Creating Information Storage Directories ... '
#
cd ..
```

```
cd info_storage
# information storage drawers
#
mkdir temp_storage
cd temp_storage
mkdir temp_doc
mkdir temp_libs
mkdir temp_stats
mkdir temp_sched
mkdir temp_data
mkdir temp_sources
mkdir temp_aux_data
chmod go+rwx *
cd ..
mkdir perm_storage
cd perm_storage
mkdir perm_doc
mkdir perm_libs
mkdir perm_stats
mkdir perm_sched
mkdir perm_data
mkdir perm_sources
mkdir perm_aux_data
chmod g+rwx *
cd ..
chmod g+rwx *
echo 'DONE'
echo -n 'Creating Individual Design Tool Directories ... '
#
cd ..
cd design_tools
#
mkdir architecture
mkdir logic
mkdir floorplan
mkdir layout
mkdir connection
mkdir topoanalysis
mkdir extraction
mkdir timing_analy
mkdir elect_analy
mkdir mask_data_gen
mkdir design_verify
mkdir design_present
mkdir utilities
chmod g+rwx *
cd ..
cd ..
echo 'DONE'
echo ' '
echo 'VLSI CAD Toolkit Directory Construction Completed.'
echo ' '
#
```

Distribution of AFIT IC CAD Tools in the Prototype VLSI CAD Toolkit

Distribution of existing CAD tools into their appropriate toolkit drawers is one of the major tasks associated with toolkit integration. This step has been performed for the AFIT CAD tools and the results are presented below. The results are presented on a drawer-by-drawer basis begiining at the top of the toolkit hierarchy. The tools are identified by using the complete UNIX pathname CURRENTLY assigned to the tools on the AFIT VAX 11/780.

DRAWER: VLSI CAD Toolkit
UNIX DIRECTORY: vlsi_tool_kit
- TOOL LIST -
custodian

DRAWER: Management Tools Drawers
UNIX DIRECTORY: mgt_tools
- TOOL LIST -

DRAWER: Configuration
UNIX DIRECTORY: configuration
- TOOL LIST -

DRAWER: Data Translation
UNIX DIRECTORY: data_transl
- TOOL LIST -
/usr/local/cad/bin/cif2ca
/usr/local/cad/bin/clipcif
/usr/local/cad/bin/clman
/usr/local/cad/bin/ca2db
/usr/local/cad/bin/ucfilt
/usr/local/cad/bin/db2cif
/usr/local/cad/bin/presim
/usr/local/cad/bin/forfor
/usr/local/cad/bin/simfilter
/usr/local/cad/bin/spcpp
/usr/local/cad/bin/rsort
/usr/local/cad/bin/window
/usr/local/cad/bin/sim2spice
/usr/local/cad/bin/pspice

DRAWER: Data Translation
UNIX DIRECTORY: data_transl
- TOOL LIST - (Contd.)
/usr/local/cad/uw/bin/netlist
/usr/local/cad/uw/bin/presim
/usr/local/cad/uw/bin/spcpp
/usr/local/cad/uw/bin/pspice
/usr/local/cad/uw/bin/ca2db
/usr/local/cad/uw/bin/simfilter
/usr/local/cad/uw/bin/spice2g6
/usr/local/cad/uw/bin/forfor
/usr/local/cad/uw/bin/prescp
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/window
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/xy2co
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/concat
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/err2co
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/blockout
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/convert
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/merge
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rplt_wd.bin
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/unconvert
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rsort
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/window
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifar
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifload
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/scifload
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/splitfile
/usr/local/cad/bin/blam
/usr/local/cad/bin/cifout-cat
/usr/local/cad/bin/convert
/usr/local/cad/bin/splitfile
/usr/local/cad/bin/unconvert
/usr/local/cad/bin/rtsyms
/usr/local/cad/bin/cifcat
/usr/local/cad/uw/bin/xsimspc

DRAWER: Resource Management
UNIX DIRECTORY: resource_mgt
- TOOL LIST -

DRAWER: Data Analysis
UNIX DIRECTORY: data_analysis
- TOOL LIST -
/usr/local/cad/bin/cifstat

DRAWER: Documentation Management
UNIX DIRECTORY: doc_mgt
- TOOL LIST -
/usr/local/cad/bin/cadman
/usr/local/cad/uw/bin/man
/usr/local/cad/uw/bin/whatis

DRAWER: Documentation Management
UNIX DIRECTORY: doc_mgt
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/whatis
/usr/local/cad/uw/man/MAKE

DRAWER: Environmental Control
UNIX DIRECTORY: environ_ctrl
- TOOL LIST -

DRAWER: Data Archival
UNIX DIRECTORY: data_archival
- TOOL LIST -
/usr/local/cad/bin/cifload
/usr/local/cad/bin/cifar
/usr/local/cad/uw/lib/libutil.a

DRAWER: Data Access Control
UNIX DIRECTORY: data_access
- TOOL LIST -

DRAWER: Data Security
UNIX DIRECTORY: data_security
- TOOL LIST -

DRAWER: Data Storage Management
UNIX DIRECTORY: data_sto_mgt
- TOOL LIST -
/usr/local/cad/bin/dir

DRAWER: Report Generation
UNIX DIRECTORY: report_gen
- TOOL LIST -

DRAWER: Standards Management
UNIX DIRECTORY: stands_mgt
- TOOL LIST -

DRAWER: Library Management
UNIX DIRECTORY: lib_mgt
- TOOL LIST -
/usr/local/cad/bin/cifload
/usr/local/cad/bin/cifar
/usr/local/cad/uw/lib/libutil.a
/usr/local/cad/lib/earl/lib_status

DRAWER: Schedule Management
UNIX DIRECTORY: sched_mgt
- TOOL LIST -

DRAWER: Design Tools Drawers
UNIX DIRECTORY: design_tools
- TOOL LIST -

DRAWER: Architectural
UNIX DIRECTORY: architecture
- TOOL LIST -

DRAWER: Logic
UNIX DIRECTORY: logic
- TOOL LIST -
/usr/local/cad/bin/plasort
/usr/local/cad/bin/eqntott
/usr/local/cad/bin/mkpla
/usr/local/cad/bin/presto
/usr/local/cad/bin/peg
/usr/local/cad/bin/plagen
/usr/local/cad/bin/plague
/usr/local/cad/bin/changeo
/usr/local/cad/bin/tpla
/usr/local/cad/bin/quilt
/usr/local/cad/bin/solve
/usr/local/cad/uw/bin/presto
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/plague
/usr/local/cad/bin/plaid
/usr/local/cad/bin/mintopla

DRAWER: Floorplan
UNIX DIRECTORY: floorplan
- TOOL LIST -

DRAWER: Layout
UNIX DIRECTORY: layout
- TOOL LIST -
/usr/local/cad/bin/bane
/usr/local/cad/bin/bane.old
/usr/local/cad/bin/plap
/usr/local/cad/bin/cll
/usr/local/cad/bin/cll2
/usr/local/cad/bin/earl
/usr/local/cad/bin/clay
/usr/local/cad/uw/bin/plap
/usr/local/cad/uw/bin/plap3

DRAWER: Layout
UNIX DIRECTORY: layout
- TOOL LIST - (Contd.)
/usr/local/cad/stanford/src/bane
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/bane

DRAWER: Interconnection
UNIX DIRECTORY: connection
- TOOL LIST -

DRAWER: Topological Analysis
UNIX DIRECTORY: topoanalysis
- TOOL LIST -
/usr/local/cad/bin/lyra
/usr/local/cad/bin/rulec
/usr/local/cad/bin/drcscript
/usr/local/cad/bin/alldrc
/usr/local/cad/bin/fastdrc
/usr/local/cad/bin/drc/bin_and
/usr/local/cad/bin/drc/bin_andnot
/usr/local/cad/bin/drc/bin_chk
/usr/local/cad/bin/drc/bin_coal
/usr/local/cad/bin/drc/bin_con
/usr/local/cad/bin/drc/bin_err
/usr/local/cad/bin/drc/bin_exp
/usr/local/cad/bin/drc/bin_filter
/usr/local/cad/bin/drc/bin_gen
/usr/local/cad/bin/drc/bin_hasnone
/usr/local/cad/bin/drc/bin_hassome
/usr/local/cad/bin/drc/bin_or
/usr/local/cad/bin/drc/ciftorect
/usr/local/cad/bin/drc/drc.back
/usr/local/cad/bin/drc/drcdoc
/usr/local/cad/bin/drc/drcscript
/usr/local/cad/bin/drc/edgetocif
/usr/local/cad/bin/drc/erc7
/usr/local/cad/bin/drc/mult
/usr/local/cad/bin/drc/oldmerge
/usr/local/cad/bin/drc/recttocif
/usr/local/cad/bin/drc/sedscript
/usr/local/cad/bin/drc/enddrc
/usr/local/cad/bin/drc/add94s
/usr/local/cad/lib/bane/drc
/usr/local/cad/lib/bane/drc/drc_filter.bak
/usr/local/cad/lib/bane/drc/drc.1
/usr/local/cad/lib/bane/drc/drc.cmos
/usr/local/cad/lib/bane/drc/drc.doc
/usr/local/cad/lib/bane/drc/drc.nmos
/usr/local/cad/lib/bane/drc/drc_filter
/usr/local/cad/lib/bane/drc/drcmask.doc
/usr/local/cad/lib/bane/drc/drcold.sa

DRAWER: Topological Analysis
UNIX DIRECTORY: topoanalysis
- TOOL LIST - (Contd.)

```
/usr/local/cad/lib/bane/drc/butconexp
/usr/local/cad/lib/bane/drc/butcondrop
/usr/local/cad/lib/bane/drc/frontsort
/usr/local/cad/lib/bane/drc/fullexp
/usr/local/cad/lib/bane/drc/impchk
/usr/local/cad/lib/bane/drc/ornmos
/usr/local/cad/lib/bane/drc/nmosextr
/usr/local/cad/lib/bane/drc/rectexp
/usr/local/cad/lib/bane/drc/rmbutcon
/usr/local/cad/lib/bane/drc/sepdpch
/usr/local/cad/lib/bane/drc/sepchk
/usr/local/cad/lib/bane/drc/sepdpbur
/usr/local/cad/lib/bane/drc/tranburied
/usr/local/cad/lib/bane/drc/trpol
/usr/local/cad/lib/bane/drc/trdiff
/usr/local/cad/lib/bane/drc/cmosextr
/usr/local/cad/lib/bane/drc/csort
/usr/local/cad/lib/bane/drc/diodechk
/usr/local/cad/lib/bane/drc/isect_exc
/usr/local/cad/lib/bane/drc/orcmos
/usr/local/cad/lib/bane/drc/pnbuttcon
/usr/local/cad/lib/bane/drc/pplusdiff
/usr/local/cad/lib/bane/drc/pwellchk
/usr/local/cad/lib/bane/drc/trandiff
/usr/local/cad/lib/bane/drc/twoint_exc
/usr/local/cad/lib/bane/drc/bool
/usr/local/cad/lib/bane/drc/boolf
/usr/local/cad/lib/bane/drc/bsort
/usr/local/cad/lib/bane/drc/conv
/usr/local/cad/lib/bane/drc/cover
/usr/local/cad/lib/bane/drc/exp
/usr/local/cad/lib/bane/drc/errout
/usr/local/cad/lib/bane/drc/expand
/usr/local/cad/lib/bane/drc/intersect
/usr/local/cad/lib/bane/drc/separate
/usr/local/cad/lib/bane/drc/sepone
/usr/local/cad/lib/bane/drc/shrink
/usr/local/cad/lib/bane/drc/tranexp
/usr/local/cad/lib/bane/drc/twocover
/usr/local/cad/bin/clldrc.lib
/usr/local/cad/bin/clldrc.lib/bool
/usr/local/cad/bin/clldrc.lib/atob
/usr/local/cad/bin/clldrc.lib/boolf
/usr/local/cad/bin/clldrc.lib/bsort
/usr/local/cad/bin/clldrc.lib/btoa
/usr/local/cad/bin/clldrc.lib/butconexp
/usr/local/cad/bin/clldrc.lib/conchk
/usr/local/cad/bin/clldrc.lib/concov
/usr/local/cad/bin/clldrc.lib/consep
/usr/local/cad/bin/clldrc.lib/conv
/usr/local/cad/bin/clldrc.lib/dbl
```

DRAWER: Topological Analysis
UNIX DIRECTORY: topoanalysis
- TOOL LIST - (Contd.)

```
/usr/local/cad/bin/clldrc.lib/drcfilter
/usr/local/cad/bin/clldrc.lib/exp
/usr/local/cad/bin/clldrc.lib/extr
/usr/local/cad/bin/clldrc.lib/frontout
/usr/local/cad/bin/clldrc.lib/frontsort
/usr/local/cad/bin/clldrc.lib/fullexp
/usr/local/cad/bin/clldrc.lib/impchk
/usr/local/cad/bin/clldrc.lib/minwidth
/usr/local/cad/bin/clldrc.lib/ordrc
/usr/local/cad/bin/clldrc.lib/pl
/usr/local/cad/bin/clldrc.lib/rectexp
/usr/local/cad/bin/clldrc.lib/rmbutcon
/usr/local/cad/bin/clldrc.lib/sepchk
/usr/local/cad/bin/clldrc.lib/sepdpch
/usr/local/cad/bin/clldrc.lib/tranexp
/usr/local/cad/bin/clldrc.lib/trpol
/usr/local/cad/bin/clldrc.lib/widchk
/usr/local/cad/uw/bin/drcscript
/usr/local/cad/uw/bin/drcpgms
/usr/local/cad/uw/bin/drcpgms/bin_and
/usr/local/cad/uw/bin/drcpgms/bin_andnot
/usr/local/cad/uw/bin/drcpgms/bin_chk
/usr/local/cad/uw/bin/drcpgms/bin_coal
/usr/local/cad/uw/bin/drcpgms/bin_con
/usr/local/cad/uw/bin/drcpgms/bin_err
/usr/local/cad/uw/bin/drcpgms/bin_exp
/usr/local/cad/uw/bin/drcpgms/bin_filter
/usr/local/cad/uw/bin/drcpgms/bin_gen
/usr/local/cad/uw/bin/drcpgms/bin_hasnone
/usr/local/cad/uw/bin/drcpgms/bin_hassome
/usr/local/cad/uw/bin/drcpgms/bin_or
/usr/local/cad/uw/bin/drcpgms/ciftorect
/usr/local/cad/uw/bin/drcpgms/recttocif
/usr/local/cad/uw/bin/drcpgms/edgetocif
/usr/local/cad/uw/bin/add94s
/usr/local/cad/uw/bin/enddrc
/usr/local/cad/uw/bin/drc
/usr/local/cad/stanford/src/DRC83/drc.cmos
/usr/local/cad/stanford/src/DRC83/drc_filter
/usr/local/cad/stanford/src/DRC83/drc.1
/usr/local/cad/stanford/src/DRC83/Makefile
/usr/local/cad/stanford/src/DRC83/drcold.sa
/usr/local/cad/stanford/src/DRC83/butconexp
/usr/local/cad/stanford/src/DRC83/butcondrop
/usr/local/cad/stanford/src/DRC83/drc.nmos
/usr/local/cad/stanford/src/DRC83/drc.doc
/usr/local/cad/stanford/src/DRC83/frontsort
/usr/local/cad/stanford/src/DRC83/fullexp
/usr/local/cad/stanford/src/DRC83/impchk
/usr/local/cad/stanford/src/DRC83/ornmos
/usr/local/cad/stanford/src/DRC83/nmosextr
```

DRAWER: Topological Analysis
UNIX DIRECTORY: topoanalysis
- TOOL LIST - (Contd.)
/usr/local/cad/stanford/src/DRC83/rectexp
/usr/local/cad/stanford/src/DRC83/rmbutcon
/usr/local/cad/stanford/src/DRC83/sepdpch
/usr/local/cad/stanford/src/DRC83/sepchk
/usr/local/cad/stanford/src/DRC83/sepdpbur
/usr/local/cad/stanford/src/DRC83/tranburied
/usr/local/cad/stanford/src/DRC83/trpol
/usr/local/cad/stanford/src/DRC83/trdiff
/usr/local/cad/stanford/src/DRC83/cmosextr
/usr/local/cad/stanford/src/DRC83/csort
/usr/local/cad/stanford/src/DRC83/diodechk
/usr/local/cad/stanford/src/DRC83/isect_exc
/usr/local/cad/stanford/src/DRC83/orcmos
/usr/local/cad/stanford/src/DRC83/pnbuttcon
/usr/local/cad/stanford/src/DRC83/pplusdiff
/usr/local/cad/stanford/src/DRC83/pwellchk
/usr/local/cad/stanford/src/DRC83/drcmask.doc
/usr/local/cad/stanford/src/DRC83/trandiff
/usr/local/cad/stanford/src/DRC83/twoint_exc
/usr/local/cad/stanford/src/DRC83/bool
/usr/local/cad/stanford/src/DRC83/boolf
/usr/local/cad/stanford/src/DRC83/bsort
/usr/local/cad/stanford/src/DRC83/conv
/usr/local/cad/stanford/src/DRC83/cover
/usr/local/cad/stanford/src/DRC83/exp
/usr/local/cad/stanford/src/DRC83/errout
/usr/local/cad/stanford/src/DRC83/expand
/usr/local/cad/stanford/src/DRC83/intersect
/usr/local/cad/stanford/src/DRC83/separate
/usr/local/cad/stanford/src/DRC83/sepone
/usr/local/cad/stanford/src/DRC83/shrink
/usr/local/cad/stanford/src/DRC83/tranexp
/usr/local/cad/stanford/src/DRC83/twocover

DRAWER: Extraction
UNIX DIRECTORY: extraction
- TOOL LIST -
/usr/local/cad/bin/mextra
/usr/local/cad/bin/netlist
/usr/local/cad/bin/cifplot

DRAWER: Timing Analysis
UNIX DIRECTORY: timing_analy
- TOOL LIST -
/usr/local/cad/bin/crystal
/usr/local/cad/bin/rnl

DRAWER: Electrical Analysis
UNIX DIRECTORY: elect_analy
- TOOL LIST -
/usr/local/cad/bin/powest
/usr/local/cad/bin/erc
/usr/local/cad/bin/ercscript
/usr/local/cad/uw/bin/erc
/usr/local/cad/uw/bin/ercscript

DRAWER: Mask Data Generation
UNIX DIRECTORY: mask_data_gen
- TOOL LIST -
/usr/local/cad/bin/cll
/usr/local/cad/bin/cll2
/usr/local/cad/bin/cifplot

DRAWER: Design Verification
UNIX DIRECTORY: design_verify
- TOOL LIST -
/usr/local/cad/bin/esim
/usr/local/cad/bin/spice2g6
/usr/local/cad/bin/spice
/usr/local/cad/bin/crystal
/usr/local/cad/bin/rnl
/usr/local/cad/uw/bin/rnl

DRAWER: Design Presentation
UNIX DIRECTORY: design_present
- TOOL LIST -
/usr/local/cad/bin/cifplot
/usr/local/cad/bin/fastcif
/usr/local/cad/bin/vdmp
/usr/local/cad/bin/print
/usr/local/cad/bin/vpltdmp
/usr/local/cad/bin/rrplot
/usr/local/cad/bin/penplot
/usr/local/cad/bin/vdump
/usr/local/cad/bin/mcp
/usr/local/cad/uw/bin/vic
/usr/local/cad/uw/bin/7221
/usr/local/cad/uw/bin/7580
/usr/local/cad/uw/bin/penplot
/usr/local/cad/uw/bin/mtp
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/aplot
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rrplot
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/tplot
/usr/local/cad/bin/listnode
/usr/local/cad/bin/mdump
/usr/local/cad/bin/7221
/usr/local/cad/bin/rrplot.bin
/usr/local/cad/bin/listesym

DRAWER: Design Presentation
UNIX DIRECTORY: design_present
- TOOL LIST - (Contd.)
/usr/local/cad/bin/mprint
/usr/local/cad/uw/bin/lspice
/usr/local/cad/bin/vlsifont
/usr/local/cad/bin/dbdi
/usr/local/cad/bin/pattern
/usr/local/cad/bin/pattern1

DRAWER: Utilities
UNIX DIRECTORY: utilities
- TOOL LIST -
/usr/local/cad/bin/uc
/usr/local/cad/bin/chksum
/usr/local/cad/bin/tvitest
/usr/local/cad/bin/inuse
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/TEST
/usr/local/cad/bin/sleeper
/usr/local/cad/bin/vprm.old
/usr/local/cad/bin/rsleeper
/usr/local/cad/bin/inter
/usr/local/cad/bin/ic
/usr/local/cad/bin/cater
/usr/local/cad/bin/xsimspc
/usr/local/cad/bin/cif
/usr/local/cad/bin/merge
/usr/local/cad/uw/bin/apropos
/usr/local/cad/bin/mas
/usr/local/cad/bin/cmem
/usr/local/cad/bin/smm
/usr/local/cad/bin/CI
/usr/local/cad/bin/smp
/usr/local/cad/bin/sme
/usr/local/cad/bin/motd
/usr/local/cad/bin/xx
/usr/local/cad/uw/bin/status
/usr/local/cad/uw/bin/pattern
/usr/local/cad/uw/bin/pattern1
/usr/local/cad/uw/bin/reset_om
/usr/local/cad/uw/bin/status_om
/usr/local/cad/uw/bin/scope
/usr/local/cad/uw/bin/pass2

DRAWER: Information Storage Drawers
UNIX DIRECTORY: info_storage
- TOOL LIST -

DRAWER: Temporary Storage Drawer
UNIX DIRECTORY: temp_storage
- TOOL LIST -

DRAWER: Documentation
UNIX DIRECTORY: temp_doc
- TOOL LIST -

DRAWER: Libraries
UNIX DIRECTORY: temp_libs
- TOOL LIST -

DRAWER: Statistics
UNIX DIRECTORY: temp_stats
- TOOL LIST -

DRAWER: Schedules
UNIX DIRECTORY: temp_sched
- TOOL LIST -

DRAWER: Design Data
UNIX DIRECTORY: temp_data
- TOOL LIST -
/usr/local/cad/sos/sos
/usr/local/cad/test/ciffile.cif
/usr/local/cad/test/pla.cll
/usr/local/cad/test/spla.cif
/usr/local/cad/test/pla.cif
/usr/local/cad/test/ciffile.drc
/usr/local/cad/test/ciffile.log
/usr/local/cad/test/pla.nodes
/usr/local/cad/test/pla.sim
/usr/local/cad/test/pla.names
/usr/local/cad/test/pla.cap
/usr/local/cad/test/pla.spice

DRAWER: Source Code
UNIX DIRECTORY: temp_sources
- TOOL LIST -

DRAWER: Aux Data
UNIX DIRECTORY: temp_aux_data
- TOOL LIST -

DRAWER: Permanent Storage Drawer
UNIX DIRECTORY: perm_storage
- TOOL LIST -

DRAWER: Documentation
UNIX DIRECTORY: perm_doc
- TOOL LIST -
/usr/local/cad/uw/doc
/usr/local/cad/uw/doc/whj.let
/usr/local/cad/uw/doc/Makefile
/usr/local/cad/uw/doc/cells
/usr/local/cad/uw/doc/READ_ME
/usr/local/cad/uw/doc/netlist
/usr/local/cad/uw/doc/plap
/usr/local/cad/uw/doc/make.out
/usr/local/cad/uw/doc/presim
/usr/local/cad/uw/doc/src
/usr/local/cad/uw/doc/src/intro.tblms
/usr/local/cad/uw/doc/src/READ_ME
/usr/local/cad/uw/doc/src/cells.tblms
/usr/local/cad/uw/doc/src/spice.tbleqnms
/usr/local/cad/uw/doc/src/READ_ME_UW
/usr/local/cad/uw/doc/src/presim.ms
/usr/local/cad/uw/doc/src/netlist.ms
/usr/local/cad/uw/doc/src/rnl.ms
/usr/local/cad/uw/doc/src/rnlbrief.ms
/usr/local/cad/uw/doc/src/plap.tblms
/usr/local/cad/uw/doc/src/plapsrc
/usr/local/cad/uw/doc/src/plapsrc/Makefile
/usr/local/cad/uw/doc/src/plapsrc/plap.0
/usr/local/cad/uw/doc/src/plapsrc/READ_ME
/usr/local/cad/uw/doc/src/plapsrc/plap.1
/usr/local/cad/uw/doc/src/plapsrc/plap.2
/usr/local/cad/uw/doc/src/plapsrc/plap.3
/usr/local/cad/uw/doc/src/plapsrc/plap.4
/usr/local/cad/uw/doc/src/plapsrc/plap.5
/usr/local/cad/uw/doc/src/plapsrc/plap.6
/usr/local/cad/uw/doc/src/plapsrc/plap.7
/usr/local/cad/uw/doc/src/plapsrc/plap.8
/usr/local/cad/uw/doc/src/plapsrc/plap.9
/usr/local/cad/uw/doc/src/plapsrc/plap.macros
/usr/local/cad/uw/doc/src/cellsrc
/usr/local/cad/uw/doc/src/cellsrc/READ_ME
/usr/local/cad/uw/doc/src/cellsrc/cellib.3
/usr/local/cad/uw/doc/src/cellsrc/Makefile
/usr/local/cad/uw/doc/src/cellsrc/cellib.0
/usr/local/cad/uw/doc/src/cellsrc/cellib.1
/usr/local/cad/uw/doc/src/cellsrc/cellib.2
/usr/local/cad/uw/doc/src/cellsrc/cellib.4
/usr/local/cad/uw/doc/src/cellsrc/cell.macros
/usr/local/cad/uw/doc/src/layout.tblms
/usr/local/cad/uw/doc/src/title.ms
/usr/local/cad/uw/doc/rnlbrief

DRAWER: Documentation
UNIX DIRECTORY: perm_doc
- TOOL LIST - (Contd.)
/usr/local/cad/uw/doc/intro
/usr/local/cad/uw/doc/layout
/usr/local/cad/uw/doc/spice
/usr/local/cad/uw/doc/whj.let.bak
/usr/local/cad/uw/doc/title
/usr/local/cad/man
/usr/local/cad/man/cat1
/usr/local/cad/man/cat1/cadman.1
/usr/local/cad/man/cat1/peg.1
/usr/local/cad/man/cat1/uc.1
/usr/local/cad/man/cat1/cifplot.1
/usr/local/cad/man/cat1/vdump.1
/usr/local/cad/man/cat1/chksum.1
/usr/local/cad/man/cat1/cif.1
/usr/local/cad/man/cat1/cif2ca.1
/usr/local/cad/man/cat1/cifar.1
/usr/local/cad/man/cat1/cifload.1
/usr/local/cad/man/cat1/cifstat.1
/usr/local/cad/man/cat1/clipcif.1
/usr/local/cad/man/cat1/cll.1
/usr/local/cad/man/cat1/clldrc.1
/usr/local/cad/man/cat1/crystal.1
/usr/local/cad/man/cat1/eqntott.1
/usr/local/cad/man/cat1/esim.1
/usr/local/cad/man/cat1/intro.1
/usr/local/cad/man/cat1/lyra.1
/usr/local/cad/man/cat1/mcp.1
/usr/local/cad/man/cat1/mextra.1
/usr/local/cad/man/cat1/mkpla.1
/usr/local/cad/man/cat1/plagen.1
/usr/local/cad/man/cat1/plague.1
/usr/local/cad/man/cat1/clay.1
/usr/local/cad/man/cat1/quilt.1
/usr/local/cad/man/cat1/rsort.1
/usr/local/cad/man/cat1/rulec.1
/usr/local/cad/man/cat1/sim2spice.1
/usr/local/cad/man/cat1/tpla.1
/usr/local/cad/man/cat1/ucfilt.1
/usr/local/cad/man/cat1/vlsifont.1
/usr/local/cad/man/cat1/window.1
/usr/local/cad/man/cat1/blam.1
/usr/local/cad/man/cat1/presto.1
/usr/local/cad/man/cat1/clman.1
/usr/local/cad/man/cat1/dir.1
/usr/local/cad/man/cat1/solve.1
/usr/local/cad/man/cat1/ar.1
/usr/local/cad/man/cat2
/usr/local/cad/man/cat3
/usr/local/cad/man/cat3/tpack.3
/usr/local/cad/man/cat4
/usr/local/cad/man/cat5

DRAWER: Documentation
UNIX DIRECTORY: perm_doc
- TOOL LIST - (Contd.)
/usr/local/cad/man/cat5/cadrc.5
/usr/local/cad/man/cat5/cifout.5
/usr/local/cad/man/cat5/intro.5
/usr/local/cad/man/cat5/tpla.5
/usr/local/cad/man/man1
/usr/local/cad/man/man1/rulec.1
/usr/local/cad/man/man1/lyra.1
/usr/local/cad/man/man1/crystal.1
/usr/local/cad/man/man1/cadman.1
/usr/local/cad/man/man1/solve.1
/usr/local/cad/man/man1/esim.1
/usr/local/cad/man/man1/vlsifont.1
/usr/local/cad/man/man1/chksum.1
/usr/local/cad/man/man1/quilt.1
/usr/local/cad/man/man1/tpla.1
/usr/local/cad/man/man1/intro.1
/usr/local/cad/man/man1/ucfilt.1
/usr/local/cad/man/man1/clipcif.1
/usr/local/cad/man/man1/cifstat.1
/usr/local/cad/man/man1/mextra.1
/usr/local/cad/man/man1/mkpla.1
/usr/local/cad/man/man1/cif2ca.1
/usr/local/cad/man/man1/mcp.1
/usr/local/cad/man/man1/cifplot.1
/usr/local/cad/man/man1/ar.1
/usr/local/cad/man/man1/peg.1
/usr/local/cad/man/man1/eqntott.1
/usr/local/cad/man/man1/sim2spice.1
/usr/local/cad/man/man1/vdump.1
/usr/local/cad/man/man1/cif.1
/usr/local/cad/man/man1/cifar.1
/usr/local/cad/man/man1/cifload.1
/usr/local/cad/man/man1/cll.1
/usr/local/cad/man/man1/clldrc.1
/usr/local/cad/man/man1/plagen.1
/usr/local/cad/man/man1/plague.1
/usr/local/cad/man/man1/rsort.1
/usr/local/cad/man/man1/window.1
/usr/local/cad/man/man1/presto.1
/usr/local/cad/man/man1/blam.1
/usr/local/cad/man/man1/dir.1
/usr/local/cad/man/man1/clman.1
/usr/local/cad/man/man1/uc.1
/usr/local/cad/man/man1/clay.1
/usr/local/cad/man/man1/cifar.sav
/usr/local/cad/man/man1/powest.1i
/usr/local/cad/man/man1/erc.1i
/usr/local/cad/man/man1/spcpp.1i
/usr/local/cad/man/man5
/usr/local/cad/man/man5/cadrc.5
/usr/local/cad/man/man5/tpla.5

DRAWER: Documentation
UNIX DIRECTORY: perm_doc
- TOOL LIST - (Contd.)
/usr/local/cad/man/man5/intro.5
/usr/local/cad/man/man5/cifout.5
/usr/local/cad/man/tmac
/usr/local/cad/man/tmac/tmac.anc.prnt
/usr/local/cad/man/tmac/ReadMe
/usr/local/cad/man/tmac/tmac.anc
/usr/local/cad/man/tmac/tmac.anc.new
/usr/local/cad/man/man3
/usr/local/cad/man/man3/tpack.3
/usr/local/cad/man/man.proto
/usr/local/cad/man/save.man
/usr/local/cad/man/man4
/usr/local/cad/uw/man
/usr/local/cad/uw/man/man1
/usr/local/cad/uw/man/man1/mexnodes.1i
/usr/local/cad/uw/man/man1/drc.1i
/usr/local/cad/uw/man/man1/erc.1i
/usr/local/cad/uw/man/man1/spice.1i
/usr/local/cad/uw/man/man1/penplot.1i
/usr/local/cad/uw/man/man1/laytools.1i
/usr/local/cad/uw/man/man1/mtp.1i
/usr/local/cad/uw/man/man1/db2cif.1i
/usr/local/cad/uw/man/man1/presim.1i
/usr/local/cad/uw/man/man1/plap.1i
/usr/local/cad/uw/man/man1/pspice.1i
/usr/local/cad/uw/man/man1/vic.1i
/usr/local/cad/uw/man/man1/simfilter.1i
/usr/local/cad/uw/man/man1/ca2db.1i
/usr/local/cad/uw/man/man1/reset_om.1
/usr/local/cad/uw/man/man1/simtools.1i
/usr/local/cad/uw/man/man1/presto.1i
/usr/local/cad/uw/man/man1/rnl.1i
/usr/local/cad/uw/man/man1/spcpp.1i
/usr/local/cad/uw/man/man1/netlist.1i
/usr/local/cad/uw/man/man1/nl.1i
/usr/local/cad/uw/man/man3
/usr/local/cad/uw/man/man3/om.3
/usr/local/cad/uw/man/man3/om_driver.3
/usr/local/cad/uw/man/man3/om_dmacros.3
/usr/local/cad/uw/man/man3/om_omacros.3
/usr/local/cad/uw/man/man3/status_om.3
/usr/local/cad/uw/man/man3/om_example.3
/usr/local/cad/uw/man/man5
/usr/local/cad/uw/man/man5/tec.5i
/usr/local/cad/uw/man/man5/simfile.5i
/usr/local/cad/uw/man/man5/technology.5i
/usr/local/cad/uw/man/Man.proto
/usr/local/cad/uw/man/READ_ME
/usr/local/cad/uw/man/cat1
/usr/local/cad/uw/man/cat1/plap.1i
/usr/local/cad/uw/man/cat1/cifplot.1i

DRAWER: Documentation
UNIX DIRECTORY: perm_doc
- TOOL LIST - (Contd.)
/usr/local/cad/uw/man/cat1/rnl.1i
/usr/local/cad/uw/man/cat1/presto.1i
/usr/local/cad/uw/man/cat1/presim.1i
/usr/local/cad/uw/man/cat1/drc.1i
/usr/local/cad/uw/man/cat1/ca2db.1i
/usr/local/cad/uw/man/cat1/mexnodes.1i
/usr/local/cad/uw/man/cat1/netlist.1i
/usr/local/cad/uw/man/cat1/penplot.1i
/usr/local/cad/uw/man/cat1/db2cif.1i
/usr/local/cad/uw/man/cat1/spice.1i
/usr/local/cad/uw/man/cat1/pspice.1i
/usr/local/cad/uw/man/cat1/laytools.1i
/usr/local/cad/uw/man/cat1/vic.1i
/usr/local/cad/uw/man/cat1/erc.1i
/usr/local/cad/uw/man/cat1/reset_om.1
/usr/local/cad/uw/man/cat1/tpla.1i
/usr/local/cad/uw/man/cat1/mtp.1i
/usr/local/cad/uw/man/cat3
/usr/local/cad/uw/man/cat3/epath.3i
/usr/local/cad/uw/man/cat5
/usr/local/cad/uw/man/cat5/tech.5i
/usr/local/cad/uw/man/cat5/tec.5i
/usr/local/cad/uw/man/cat5/technology.5i
/usr/local/cad/uw/man/cat5/simfile.5i
/usr/local/cad/uw/CONTENTS
/usr/local/cad/uw/README
/usr/local/cad/uw/directory
/usr/local/cad/uw/lib/tmac/VLSIman.tmac
/usr/local/cad/lib/earl/cpads.doc
/usr/local/cad/lib/earl/npads.doc
/usr/local/cad/stanford/src/DRC83/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifar.1
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/DOC
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/READ_ME
/usr/local/cad/stanford/src/PLAGEN/PLA/pla.1
/usr/local/cad/stanford/src/PLAGEN/PLA/lex.1
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/plague.1
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/plague.1

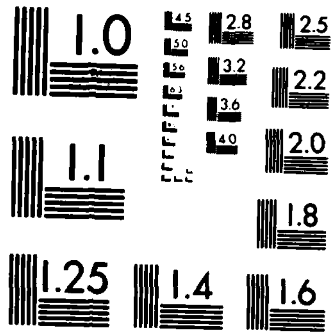
DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST -
/usr/local/cad/lib
/usr/local/cad/lib/s_ext.cll
/usr/local/cad/mosis

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

```
/usr/local/cad/mosis/40p69x68.cif
/usr/local/cad/mosis/40p69x68.list
/usr/local/cad/mosis/64p69x68.cif
/usr/local/cad/mosis/64p69x68.list
/usr/local/cad/lib/lyra
/usr/local/cad/lib/lyra/nmosBERK.r
/usr/local/cad/lib/lyra/cmos-pwJPL.r
/usr/local/cad/lib/lyra/nmosBERK
/usr/local/cad/lib/lyra/Lyra.proto
/usr/local/cad/lib/lyra/nmosMC.r
/usr/local/cad/lib/lyra/Rulecl
/usr/local/cad/lib/lyra/DEFAULTS
/usr/local/cad/lib/tpla
/usr/local/cad/lib/tpla/Makefile
/usr/local/cad/lib/tpla/p-Tcis.tp
/usr/local/cad/lib/tpla/p-Mtrans.tp
/usr/local/cad/lib/tpla/p-Mcis.tp
/usr/local/cad/lib/tpla/p-Btrans.tp
/usr/local/cad/lib/tpla/p-Bcis.tp
/usr/local/cad/lib/tpla/ReadMe
/usr/local/cad/lib/tpla/p.tp
/usr/local/cad/lib/tpla/p-Ttrans.tp
/usr/local/cad/lib/quilt
/usr/local/cad/lib/quilt/q-vlsifont.tp
/usr/local/cad/lib/slang
/usr/local/cad/lib/slang/simmacs.l
/usr/local/cad/lib/slang/try3.l
/usr/local/cad/lib/slang/try2.l
/usr/local/cad/lib/slang/try1.l
/usr/local/cad/lib/slang/sproject.l
/usr/local/cad/lib/slang/simscr.l
/usr/local/cad/lib/slang/jkfmacs.l
/usr/local/cad/lib/slang/slang.spec.h
/usr/local/cad/lib/ext.cll
/usr/local/cad/lib/displays
/usr/local/cad/lib/tpack.lib
/usr/local/cad/lib/tpack.h
/usr/local/cad/lib/extname
/usr/local/cad/lib/tfill
/usr/local/cad/lib/ufill
/usr/local/cad/lib/vfill
/usr/local/cad/lib/hp
/usr/local/cad/lib/trilog
/usr/local/cad/lib/mcpl
/usr/local/cad/lib/cifwindow
/usr/local/cad/lib/cifflatten
/usr/local/cad/lib/fix.6
/usr/local/cad/lib/pat.unknown
/usr/local/cad/lib/pat.nmos
/usr/local/cad/lib/pat.jpl
/usr/local/cad/lib/pat.cmos.info
```

AD-A151 813 THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE 3/3
INTEGRATED CIRCUIT COMPUTE..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED T R VERMILLION DEC 84 AFIT/GE/ENG/84D-68 F/G 9/2 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/mosis/40p69x68.cif
/usr/local/cad/mosis/40p69x68.list
/usr/local/cad/mosis/64p69x68.cif
/usr/local/cad/mosis/64p69x68.list
/usr/local/cad/lib/lyra
/usr/local/cad/lib/lyra/nmosBERK.r
/usr/local/cad/lib/lyra/cmos-pwJPL.r
/usr/local/cad/lib/lyra/nmosBERK
/usr/local/cad/lib/lyra/Lyra.proto
/usr/local/cad/lib/lyra/nmosMC.r
/usr/local/cad/lib/lyra/Rulec1
/usr/local/cad/lib/lyra/DEFAULTS
/usr/local/cad/lib/tpla
/usr/local/cad/lib/tpla/Makefile
/usr/local/cad/lib/tpla/p-Tcis.tp
/usr/local/cad/lib/tpla/p-Mtrans.tp
/usr/local/cad/lib/tpla/p-Mcis.tp
/usr/local/cad/lib/tpla/p-Btrans.tp
/usr/local/cad/lib/tpla/p-Bcis.tp
/usr/local/cad/lib/tpla/ReadMe
/usr/local/cad/lib/tpla/p.tp
/usr/local/cad/lib/tpla/p-Ttrans.tp
/usr/local/cad/lib/quilt
/usr/local/cad/lib/quilt/q-vlsifont.tp
/usr/local/cad/lib/slang
/usr/local/cad/lib/slang/simmacs.l
/usr/local/cad/lib/slang/try3.l
/usr/local/cad/lib/slang/try2.l
/usr/local/cad/lib/slang/try1.l
/usr/local/cad/lib/slang/sproject.l
/usr/local/cad/lib/slang/simscr.l
/usr/local/cad/lib/slang/jkfmacs.l
/usr/local/cad/lib/slang/slang.spec.h
/usr/local/cad/lib/ext.cll
/usr/local/cad/lib/displays
/usr/local/cad/lib/tpack.lib
/usr/local/cad/lib/tpack.h
/usr/local/cad/lib/extname
/usr/local/cad/lib/tfill
/usr/local/cad/lib/ufill
/usr/local/cad/lib/vfill
/usr/local/cad/lib/hp
/usr/local/cad/lib/trilog
/usr/local/cad/lib/mcp1
/usr/local/cad/lib/cifwindow
/usr/local/cad/lib/cifflatten
/usr/local/cad/lib/fix.6
/usr/local/cad/lib/pat.unknown
/usr/local/cad/lib/pat.nmos
/usr/local/cad/lib/pat.jpl
/usr/local/cad/lib/pat.cmos.info

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/lib/pat.cmos
/usr/local/cad/lib/extract
/usr/local/cad/lib/extpass3
/usr/local/cad/lib/PlaStaticIn.ne
/usr/local/cad/lib/log
/usr/local/cad/lib/libs.cif
/usr/local/cad/lib/libs.lib
/usr/local/cad/lib/libplap.a
/usr/local/cad/lib/tmac
/usr/local/cad/lib/pshell.p
/usr/local/cad/lib/s_ext.cll.old
/usr/local/cad/lib/save.tmp
/usr/local/cad/lib/libs.lib.new
/usr/local/cad/lib/PlaStaticIn.ol
/usr/local/cad/uw/lib/cellib
/usr/local/cad/uw/lib/cellib/nmos
/usr/local/cad/uw/lib/cellib/nmos/28p23x34.att
/usr/local/cad/uw/lib/cellib/nmos/28p23x34.ca
/usr/local/cad/uw/lib/cellib/nmos/28p23x34.sym
/usr/local/cad/uw/lib/cellib/nmos/28p46x34.att
/usr/local/cad/uw/lib/cellib/nmos/28p46x34.ca
/usr/local/cad/uw/lib/cellib/nmos/28p46x34.sym
/usr/local/cad/uw/lib/cellib/nmos/40p46x34.att
/usr/local/cad/uw/lib/cellib/nmos/40p46x34.ca
/usr/local/cad/uw/lib/cellib/nmos/40p46x34.sym
/usr/local/cad/uw/lib/cellib/nmos/40p46x68.att
/usr/local/cad/uw/lib/cellib/nmos/40p46x68.ca
/usr/local/cad/uw/lib/cellib/nmos/40p46x68.sym
/usr/local/cad/uw/lib/cellib/nmos/40p69x68.att
/usr/local/cad/uw/lib/cellib/nmos/40p69x68.ca
/usr/local/cad/uw/lib/cellib/nmos/40p69x68.sym
/usr/local/cad/uw/lib/cellib/nmos/64p46x68.att
/usr/local/cad/uw/lib/cellib/nmos/64p46x68.ca
/usr/local/cad/uw/lib/cellib/nmos/64p46x68.sym
/usr/local/cad/uw/lib/cellib/nmos/64p69x68.att
/usr/local/cad/uw/lib/cellib/nmos/64p69x68.ca
/usr/local/cad/uw/lib/cellib/nmos/64p69x68.sym
/usr/local/cad/uw/lib/cellib/nmos/64p79x92.att
/usr/local/cad/uw/lib/cellib/nmos/64p79x92.ca
/usr/local/cad/uw/lib/cellib/nmos/64p79x92.sym
/usr/local/cad/uw/lib/cellib/nmos/84p69x68.att
/usr/local/cad/uw/lib/cellib/nmos/84p69x68.ca
/usr/local/cad/uw/lib/cellib/nmos/84p69x68.sym
/usr/local/cad/uw/lib/cellib/nmos/84p79x92.att
/usr/local/cad/uw/lib/cellib/nmos/84p79x92.ca
/usr/local/cad/uw/lib/cellib/nmos/84p79x92.sym
/usr/local/cad/uw/lib/cellib/nmos/ClkLogic.att
/usr/local/cad/uw/lib/cellib/nmos/ClkLogic.ca
/usr/local/cad/uw/lib/cellib/nmos/ClkLogic.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAbin.att
/usr/local/cad/uw/lib/cellib/nmos/PLAbin.ca

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/cellib/nmos/PLAbin.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAbout.att
/usr/local/cad/uw/lib/cellib/nmos/PLAbout.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAbout.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAbpllp.att
/usr/local/cad/uw/lib/cellib/nmos/PLAbpllp.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAbpllp.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAcell.att
/usr/local/cad/uw/lib/cellib/nmos/PLAcell.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAcell.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAcon.att
/usr/local/cad/uw/lib/cellib/nmos/PLAcon.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAcon.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAground.att
/usr/local/cad/uw/lib/cellib/nmos/PLAground.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAground.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAin.att
/usr/local/cad/uw/lib/cellib/nmos/PLAin.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAin.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAjog.att
/usr/local/cad/uw/lib/cellib/nmos/PLAjog.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAjog.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAmxjog.att
/usr/local/cad/uw/lib/cellib/nmos/PLAmxjog.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAmxjog.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAout.att
/usr/local/cad/uw/lib/cellib/nmos/PLAout.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAout.sym
/usr/local/cad/uw/lib/cellib/nmos/PLApllp.att
/usr/local/cad/uw/lib/cellib/nmos/PLApllp.ca
/usr/local/cad/uw/lib/cellib/nmos/PLApllp.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAspace.att
/usr/local/cad/uw/lib/cellib/nmos/PLAspace.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAspace.sym
/usr/local/cad/uw/lib/cellib/nmos/PLAtee.att
/usr/local/cad/uw/lib/cellib/nmos/PLAtee.ca
/usr/local/cad/uw/lib/cellib/nmos/PLAtee.sym
/usr/local/cad/uw/lib/cellib/nmos/Pad3State.att
/usr/local/cad/uw/lib/cellib/nmos/Pad3State.ca
/usr/local/cad/uw/lib/cellib/nmos/Pad3State.sym
/usr/local/cad/uw/lib/cellib/nmos/PadBlank.att
/usr/local/cad/uw/lib/cellib/nmos/PadBlank.ca
/usr/local/cad/uw/lib/cellib/nmos/PadBlank.sym
/usr/local/cad/uw/lib/cellib/nmos/PadClkBar.att
/usr/local/cad/uw/lib/cellib/nmos/PadClkBar.ca
/usr/local/cad/uw/lib/cellib/nmos/PadClkBar.sym
/usr/local/cad/uw/lib/cellib/nmos/PadClkO.att
/usr/local/cad/uw/lib/cellib/nmos/PadClkO.ca
/usr/local/cad/uw/lib/cellib/nmos/PadClkO.sym
/usr/local/cad/uw/lib/cellib/nmos/PadDriver.att
/usr/local/cad/uw/lib/cellib/nmos/PadDriver.ca

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/cellib/nmos/PadDriver.sym
/usr/local/cad/uw/lib/cellib/nmos/PadGround.att
/usr/local/cad/uw/lib/cellib/nmos/PadGround.ca
/usr/local/cad/uw/lib/cellib/nmos/PadGround.sym
/usr/local/cad/uw/lib/cellib/nmos/PadIn.att
/usr/local/cad/uw/lib/cellib/nmos/PadIn.ca
/usr/local/cad/uw/lib/cellib/nmos/PadIn.sym
/usr/local/cad/uw/lib/cellib/nmos/PadOut.att
/usr/local/cad/uw/lib/cellib/nmos/PadOut.ca
/usr/local/cad/uw/lib/cellib/nmos/PadOut.sym
/usr/local/cad/uw/lib/cellib/nmos/PadVdd.att
/usr/local/cad/uw/lib/cellib/nmos/PadVdd.ca
/usr/local/cad/uw/lib/cellib/nmos/PadVdd.sym
/usr/local/cad/uw/lib/cellib/nmos/gb.att
/usr/local/cad/uw/lib/cellib/nmos/gb.ca
/usr/local/cad/uw/lib/cellib/nmos/gb.sym
/usr/local/cad/uw/lib/cellib/nmos/greast.att
/usr/local/cad/uw/lib/cellib/nmos/greast.ca
/usr/local/cad/uw/lib/cellib/nmos/greast.sym
/usr/local/cad/uw/lib/cellib/nmos/makefile
/usr/local/cad/uw/lib/cellib/nmos/nmos.doc
/usr/local/cad/uw/lib/cellib/nmos/pulldownE.att
/usr/local/cad/uw/lib/cellib/nmos/pulldownE.ca
/usr/local/cad/uw/lib/cellib/nmos/pulldownE.sym
/usr/local/cad/uw/lib/cellib/nmos/pulldownN.att
/usr/local/cad/uw/lib/cellib/nmos/pulldownN.ca
/usr/local/cad/uw/lib/cellib/nmos/pulldownN.sym
/usr/local/cad/uw/lib/cellib/nmos/pulldownS.att
/usr/local/cad/uw/lib/cellib/nmos/pulldownS.ca
/usr/local/cad/uw/lib/cellib/nmos/pulldownS.sym
/usr/local/cad/uw/lib/cellib/nmos/pulldownW.att
/usr/local/cad/uw/lib/cellib/nmos/pulldownW.ca
/usr/local/cad/uw/lib/cellib/nmos/pulldownW.sym
/usr/local/cad/uw/lib/cellib/nmos/rb.att
/usr/local/cad/uw/lib/cellib/nmos/rb.ca
/usr/local/cad/uw/lib/cellib/nmos/rb.sym
/usr/local/cad/uw/lib/cellib/nmos/symbol1.att
/usr/local/cad/uw/lib/cellib/nmos/symbol1.ca
/usr/local/cad/uw/lib/cellib/nmos/symbol1.sym
/usr/local/cad/uw/lib/cellib/nmos/symbol3.att
/usr/local/cad/uw/lib/cellib/nmos/symbol3.ca
/usr/local/cad/uw/lib/cellib/nmos/symbol3.sym
/usr/local/cad/uw/lib/cellib/cmospw
/usr/local/cad/uw/lib/cellib/cmospw/28p46x34.att
/usr/local/cad/uw/lib/cellib/cmospw/28p46x34.ca
/usr/local/cad/uw/lib/cellib/cmospw/28p46x34.sym
/usr/local/cad/uw/lib/cellib/cmospw/2binttlbuf.att
/usr/local/cad/uw/lib/cellib/cmospw/2binttlbuf.ca
/usr/local/cad/uw/lib/cellib/cmospw/2binttlbuf.sym
/usr/local/cad/uw/lib/cellib/cmospw/40p46x68.att
/usr/local/cad/uw/lib/cellib/cmospw/40p46x68.ca

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

```
/usr/local/cad/uw/lib/cellib/cmospw/40p46x68.sym
/usr/local/cad/uw/lib/cellib/cmospw/40p69x68.att
/usr/local/cad/uw/lib/cellib/cmospw/40p69x68.ca
/usr/local/cad/uw/lib/cellib/cmospw/40p69x68.sym
/usr/local/cad/uw/lib/cellib/cmospw/64p69x68.att
/usr/local/cad/uw/lib/cellib/cmospw/64p69x68.ca
/usr/local/cad/uw/lib/cellib/cmospw/64p69x68.sym
/usr/local/cad/uw/lib/cellib/cmospw/64p79x92.att
/usr/local/cad/uw/lib/cellib/cmospw/64p79x92.ca
/usr/local/cad/uw/lib/cellib/cmospw/64p79x92.sym
/usr/local/cad/uw/lib/cellib/cmospw/84p79x92.att
/usr/local/cad/uw/lib/cellib/cmospw/84p79x92.ca
/usr/local/cad/uw/lib/cellib/cmospw/84p79x92.sym
/usr/local/cad/uw/lib/cellib/cmospw/_.att
/usr/local/cad/uw/lib/cellib/cmospw/_.ca
/usr/local/cad/uw/lib/cellib/cmospw/_.sym
/usr/local/cad/uw/lib/cellib/cmospw/__.att
/usr/local/cad/uw/lib/cellib/cmospw/__.ca
/usr/local/cad/uw/lib/cellib/cmospw/__.sym
/usr/local/cad/uw/lib/cellib/cmospw/_arrest.att
/usr/local/cad/uw/lib/cellib/cmospw/_arrest.ca
/usr/local/cad/uw/lib/cellib/cmospw/_arrest.sym
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf.att
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf.ca
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf.sym
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf2.att
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf2.ca
/usr/local/cad/uw/lib/cellib/cmospw/_bin-buf2.sym
/usr/local/cad/uw/lib/cellib/cmospw/_dm.att
/usr/local/cad/uw/lib/cellib/cmospw/_dm.ca
/usr/local/cad/uw/lib/cellib/cmospw/_dm.sym
/usr/local/cad/uw/lib/cellib/cmospw/_driveh.att
/usr/local/cad/uw/lib/cellib/cmospw/_driveh.ca
/usr/local/cad/uw/lib/cellib/cmospw/_driveh.sym
/usr/local/cad/uw/lib/cellib/cmospw/_drivel.att
/usr/local/cad/uw/lib/cellib/cmospw/_drivel.ca
/usr/local/cad/uw/lib/cellib/cmospw/_drivel.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate1.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate1.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate1.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate2.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate2.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate2.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate3.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate3.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate3.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate4.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate4.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate4.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate5.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate5.ca
```

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

```
/usr/local/cad/uw/lib/cellib/cmospw/_gate5.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate7.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate7.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate7.sym
/usr/local/cad/uw/lib/cellib/cmospw/_gate8.att
/usr/local/cad/uw/lib/cellib/cmospw/_gate8.ca
/usr/local/cad/uw/lib/cellib/cmospw/_gate8.sym
/usr/local/cad/uw/lib/cellib/cmospw/_inpro.att
/usr/local/cad/uw/lib/cellib/cmospw/_inpro.ca
/usr/local/cad/uw/lib/cellib/cmospw/_inpro.sym
/usr/local/cad/uw/lib/cellib/cmospw/_out-buf.att
/usr/local/cad/uw/lib/cellib/cmospw/_out-buf.ca
/usr/local/cad/uw/lib/cellib/cmospw/_out-buf.sym
/usr/local/cad/uw/lib/cellib/cmospw/_padpart.att
/usr/local/cad/uw/lib/cellib/cmospw/_padpart.ca
/usr/local/cad/uw/lib/cellib/cmospw/_padpart.sym
/usr/local/cad/uw/lib/cellib/cmospw/_plug.att
/usr/local/cad/uw/lib/cellib/cmospw/_plug.ca
/usr/local/cad/uw/lib/cellib/cmospw/_plug.sym
/usr/local/cad/uw/lib/cellib/cmospw/_pm.att
/usr/local/cad/uw/lib/cellib/cmospw/_pm.ca
/usr/local/cad/uw/lib/cellib/cmospw/_pm.sym
/usr/local/cad/uw/lib/cellib/cmospw/_pm2.att
/usr/local/cad/uw/lib/cellib/cmospw/_pm2.ca
/usr/local/cad/uw/lib/cellib/cmospw/_pm2.sym
/usr/local/cad/uw/lib/cellib/cmospw/_pm3.att
/usr/local/cad/uw/lib/cellib/cmospw/_pm3.ca
/usr/local/cad/uw/lib/cellib/cmospw/_pm3.sym
/usr/local/cad/uw/lib/cellib/cmospw/_pwt.att
/usr/local/cad/uw/lib/cellib/cmospw/_pwt.ca
/usr/local/cad/uw/lib/cellib/cmospw/_pwt.sym
/usr/local/cad/uw/lib/cellib/cmospw/_resistor.att
/usr/local/cad/uw/lib/cellib/cmospw/_resistor.ca
/usr/local/cad/uw/lib/cellib/cmospw/_resistor.sym
/usr/local/cad/uw/lib/cellib/cmospw/_tri-buf.att
/usr/local/cad/uw/lib/cellib/cmospw/_tri-buf.ca
/usr/local/cad/uw/lib/cellib/cmospw/_tri-buf.sym
/usr/local/cad/uw/lib/cellib/cmospw/_ttl-driveh.at
/usr/local/cad/uw/lib/cellib/cmospw/_ttl-driveh.ca
/usr/local/cad/uw/lib/cellib/cmospw/_ttl-driveh.sy
/usr/local/cad/uw/lib/cellib/cmospw/binttlbuf.att
/usr/local/cad/uw/lib/cellib/cmospw/binttlbuf.ca
/usr/local/cad/uw/lib/cellib/cmospw/binttlbuf.sym
/usr/local/cad/uw/lib/cellib/cmospw/gb.att
/usr/local/cad/uw/lib/cellib/cmospw/gb.ca
/usr/local/cad/uw/lib/cellib/cmospw/gb.sym
/usr/local/cad/uw/lib/cellib/cmospw/makefile
/usr/local/cad/uw/lib/cellib/cmospw/outttlbuf.att
/usr/local/cad/uw/lib/cellib/cmospw/outttlbuf.ca
/usr/local/cad/uw/lib/cellib/cmospw/outttlbuf.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1.att
```

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/cellib/cmospw/pad1.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin-ttl.at
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin-ttl.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin-ttl.sy
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1bin.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1gnd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1gnd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1gnd.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1in.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1in.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1in.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1out-ttl.at
/usr/local/cad/uw/lib/cellib/cmospw/pad1out-ttl.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1out-ttl.sy
/usr/local/cad/uw/lib/cellib/cmospw/pad1out.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1out.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1out.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1space.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1space.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1space.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1ts.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1ts.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1ts.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad1vdd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad1vdd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad1vdd.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin-ttl.at
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin-ttl.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin-ttl.sy
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2bin.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2gnd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2gnd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2gnd.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2in.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2in.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2in.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2space.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2space.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2space.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad2vdd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad2vdd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad2vdd.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad3.att

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

```
/usr/local/cad/uw/lib/cellib/cmospw/pad3.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad3.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad3gnd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad3gnd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad3gnd.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad3in.att
/usr/local/cad/uw/lib/cellib/cmospw/pad3in.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad3in.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad3space.att
/usr/local/cad/uw/lib/cellib/cmospw/pad3space.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad3space.sym
/usr/local/cad/uw/lib/cellib/cmospw/pad3vdd.att
/usr/local/cad/uw/lib/cellib/cmospw/pad3vdd.ca
/usr/local/cad/uw/lib/cellib/cmospw/pad3vdd.sym
/usr/local/cad/uw/lib/cellib/cmospw/project.att
/usr/local/cad/uw/lib/cellib/cmospw/project.ca
/usr/local/cad/uw/lib/cellib/cmospw/project.sym
/usr/local/cad/uw/lib/cellib/cmospw/rb.att
/usr/local/cad/uw/lib/cellib/cmospw/rb.ca
/usr/local/cad/uw/lib/cellib/cmospw/rb.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_0.att
/usr/local/cad/uw/lib/cellib/cmospw/s_0.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_0.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_1.att
/usr/local/cad/uw/lib/cellib/cmospw/s_1.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_1.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_10.att
/usr/local/cad/uw/lib/cellib/cmospw/s_10.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_10.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_2.att
/usr/local/cad/uw/lib/cellib/cmospw/s_2.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_2.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_3.att
/usr/local/cad/uw/lib/cellib/cmospw/s_3.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_3.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_4.att
/usr/local/cad/uw/lib/cellib/cmospw/s_4.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_4.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_5.att
/usr/local/cad/uw/lib/cellib/cmospw/s_5.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_5.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_6.att
/usr/local/cad/uw/lib/cellib/cmospw/s_6.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_6.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_7.att
/usr/local/cad/uw/lib/cellib/cmospw/s_7.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_7.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_8.att
/usr/local/cad/uw/lib/cellib/cmospw/s_8.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_8.sym
/usr/local/cad/uw/lib/cellib/cmospw/s_9.att
```

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

```
/usr/local/cad/uw/lib/cellib/cmospw/s_9.ca
/usr/local/cad/uw/lib/cellib/cmospw/s_9.sym
/usr/local/cad/uw/lib/cellib/isocmos
/usr/local/cad/uw/lib/cellib/isocmos/README
/usr/local/cad/uw/lib/cellib/isocmos/clkinv.att
/usr/local/cad/uw/lib/cellib/isocmos/clkinv.ca
/usr/local/cad/uw/lib/cellib/isocmos/clkinv.sym
/usr/local/cad/uw/lib/cellib/isocmos/dlatch.att
/usr/local/cad/uw/lib/cellib/isocmos/dlatch.ca
/usr/local/cad/uw/lib/cellib/isocmos/dlatch.sym
/usr/local/cad/uw/lib/cellib/isocmos/dlatchr.att
/usr/local/cad/uw/lib/cellib/isocmos/dlatchr.ca
/usr/local/cad/uw/lib/cellib/isocmos/dlatchr.sym
/usr/local/cad/uw/lib/cellib/isocmos/dp.att
/usr/local/cad/uw/lib/cellib/isocmos/dp.ca
/usr/local/cad/uw/lib/cellib/isocmos/dp.sym
/usr/local/cad/uw/lib/cellib/isocmos/frame40.att
/usr/local/cad/uw/lib/cellib/isocmos/frame40.ca
/usr/local/cad/uw/lib/cellib/isocmos/frame40.sym
/usr/local/cad/uw/lib/cellib/isocmos/gb.att
/usr/local/cad/uw/lib/cellib/isocmos/gb.ca
/usr/local/cad/uw/lib/cellib/isocmos/gb.sym
/usr/local/cad/uw/lib/cellib/isocmos/gnd.att
/usr/local/cad/uw/lib/cellib/isocmos/gnd.ca
/usr/local/cad/uw/lib/cellib/isocmos/gnd.sym
/usr/local/cad/uw/lib/cellib/isocmos/inpad.att
/usr/local/cad/uw/lib/cellib/isocmos/inpad.ca
/usr/local/cad/uw/lib/cellib/isocmos/inpad.sym
/usr/local/cad/uw/lib/cellib/isocmos/inv.att
/usr/local/cad/uw/lib/cellib/isocmos/inv.ca
/usr/local/cad/uw/lib/cellib/isocmos/inv.sym
/usr/local/cad/uw/lib/cellib/isocmos/makefile
/usr/local/cad/uw/lib/cellib/isocmos/md.att
/usr/local/cad/uw/lib/cellib/isocmos/md.ca
/usr/local/cad/uw/lib/cellib/isocmos/md.sym
/usr/local/cad/uw/lib/cellib/isocmos/mp.att
/usr/local/cad/uw/lib/cellib/isocmos/mp.ca
/usr/local/cad/uw/lib/cellib/isocmos/mp.sym
/usr/local/cad/uw/lib/cellib/isocmos/nand2.att
/usr/local/cad/uw/lib/cellib/isocmos/nand2.ca
/usr/local/cad/uw/lib/cellib/isocmos/nand2.sym
/usr/local/cad/uw/lib/cellib/isocmos/nand3.att
/usr/local/cad/uw/lib/cellib/isocmos/nand3.ca
/usr/local/cad/uw/lib/cellib/isocmos/nand3.sym
/usr/local/cad/uw/lib/cellib/isocmos/nand4.att
/usr/local/cad/uw/lib/cellib/isocmos/nand4.ca
/usr/local/cad/uw/lib/cellib/isocmos/nand4.sym
/usr/local/cad/uw/lib/cellib/isocmos/nor2.att
/usr/local/cad/uw/lib/cellib/isocmos/nor2.ca
/usr/local/cad/uw/lib/cellib/isocmos/nor2.sym
/usr/local/cad/uw/lib/cellib/isocmos/nor3.att
```

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/cellib/isocmos/nor3.ca
/usr/local/cad/uw/lib/cellib/isocmos/nor3.sym
/usr/local/cad/uw/lib/cellib/isocmos/nor4.att
/usr/local/cad/uw/lib/cellib/isocmos/nor4.ca
/usr/local/cad/uw/lib/cellib/isocmos/nor4.sym
/usr/local/cad/uw/lib/cellib/isocmos/on.att
/usr/local/cad/uw/lib/cellib/isocmos/on.ca
/usr/local/cad/uw/lib/cellib/isocmos/on.sym
/usr/local/cad/uw/lib/cellib/isocmos/op.att
/usr/local/cad/uw/lib/cellib/isocmos/op.ca
/usr/local/cad/uw/lib/cellib/isocmos/op.sym
/usr/local/cad/uw/lib/cellib/isocmos/outpad.att
/usr/local/cad/uw/lib/cellib/isocmos/outpad.ca
/usr/local/cad/uw/lib/cellib/isocmos/outpad.sym
/usr/local/cad/uw/lib/cellib/isocmos/pvs.att
/usr/local/cad/uw/lib/cellib/isocmos/pvs.ca
/usr/local/cad/uw/lib/cellib/isocmos/pvs.sym
/usr/local/cad/uw/lib/cellib/isocmos/rb.att
/usr/local/cad/uw/lib/cellib/isocmos/rb.ca
/usr/local/cad/uw/lib/cellib/isocmos/rb.sym
/usr/local/cad/uw/lib/cellib/isocmos/sel2inv.att
/usr/local/cad/uw/lib/cellib/isocmos/sel2inv.ca
/usr/local/cad/uw/lib/cellib/isocmos/sel2inv.sym
/usr/local/cad/uw/lib/cellib/isocmos/svd.att
/usr/local/cad/uw/lib/cellib/isocmos/svd.ca
/usr/local/cad/uw/lib/cellib/isocmos/svd.sym
/usr/local/cad/uw/lib/cellib/isocmos/tripad.att
/usr/local/cad/uw/lib/cellib/isocmos/tripad.ca
/usr/local/cad/uw/lib/cellib/isocmos/tripad.sym
/usr/local/cad/uw/lib/cellib/isocmos/tripadm.att
/usr/local/cad/uw/lib/cellib/isocmos/tripadm.ca
/usr/local/cad/uw/lib/cellib/isocmos/tripadm.sym
/usr/local/cad/uw/lib/cellib/isocmos/vdd.att
/usr/local/cad/uw/lib/cellib/isocmos/vdd.ca
/usr/local/cad/uw/lib/cellib/isocmos/vdd.sym
/usr/local/cad/uw/lib/cellib/isocmos/xor2.att
/usr/local/cad/uw/lib/cellib/isocmos/xor2.ca
/usr/local/cad/uw/lib/cellib/isocmos/xor2.sym
/usr/local/cad/uw/lib/rnl/plot.l
/usr/local/cad/uw/lib/rnl/uwsim.l
/usr/local/cad/uw/lib/rnl/uwstd.l
/usr/local/cad/lib/bane/libc.lib
/usr/local/cad/lib/bane/inv.cll
/usr/local/cad/lib/bane/libacc.a
/usr/local/cad/lib/bane/liblocal.a
/usr/local/cad/lib/bane/stipples.c
/usr/local/cad/lib/bane/stipples.h
/usr/local/cad/lib/bane/c_ext.cll
/usr/local/cad/lib/bane/stipples.x
/usr/local/cad/lib/bane/aedcolors.h
/usr/local/cad/lib/bane/cllmap.cmos

DRAWER: Libraries
UNIX DIRECTORY: perm_libs
- TOOL LIST - (Contd.)

/usr/local/cad/lib/bane/cllmap.nmos
/usr/local/cad/lib/bane/cllmap.pmos
/usr/local/cad/lib/bane/s_ext.cll
/usr/local/cad/lib/bane/libs.cif
/usr/local/cad/lib/bane/cifmap.cmos
/usr/local/cad/lib/bane/cifmap.nmos
/usr/local/cad/lib/bane/cifmap.dmos
/usr/local/cad/lib/bane/cifmap.pmos
/usr/local/cad/lib/bane/font.h
/usr/local/cad/lib/bane/font.GouldB
/usr/local/cad/lib/bane/libs.lib
/usr/local/cad/lib/bane/READ_ME
/usr/local/cad/lib/clay/ciflib/H_bothdiff
/usr/local/cad/lib/clay/ciflib/V_bothdiff
/usr/local/cad/lib/clay/ciflib/bottomdiff
/usr/local/cad/lib/clay/ciflib/diff
/usr/local/cad/lib/clay/ciflib/dpcontact
/usr/local/cad/lib/clay/ciflib/dpcontacthd
/usr/local/cad/lib/clay/ciflib/dpcontactvd
/usr/local/cad/lib/clay/ciflib/leftdiff
/usr/local/cad/lib/clay/ciflib/mdcontact
/usr/local/cad/lib/clay/ciflib/metal
/usr/local/cad/lib/clay/ciflib/mpcontact
/usr/local/cad/lib/clay/ciflib/poly
/usr/local/cad/lib/clay/ciflib/pulldown
/usr/local/cad/lib/clay/ciflib/pulldownr
/usr/local/cad/lib/clay/ciflib/pullup
/usr/local/cad/lib/clay/ciflib/pullupvg
/usr/local/cad/lib/clay/ciflib/rightdiff
/usr/local/cad/lib/clay/ciflib/topdiff
/usr/local/cad/lib/earl/CMOS.head
/usr/local/cad/lib/earl/CMOS.head.cif
/usr/local/cad/lib/earl/NMOS.head
/usr/local/cad/lib/earl/NMOS.head.old
/usr/local/cad/lib/earl/clap
/usr/local/cad/lib/earl/cmos
/usr/local/cad/lib/earl/cpads.e
/usr/local/cad/lib/earl/nmos
/usr/local/cad/lib/earl/npadexamp.e
/usr/local/cad/lib/earl/npads.e
/usr/local/cad/lib/earl/old
/usr/local/cad/lib/earl/outpad.cif
/usr/local/cad/lib/earl/patterns
/usr/local/cad/lib/earl/river.e
/usr/local/cad/lib/earl/sospads.e
/usr/local/cad/lib/earl/test.head
/usr/local/cad/lib/earl/tripad.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/CMOS
/pmostest.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/CMOS
/RESULTS/pmostest.co

```
DRAWER:  Libraries
UNIX DIRECTORY:  perm_libs
- TOOL LIST -  (Contd.)
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/CMOS
                                    /RESULTS/cmostest.co
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/CMOS
                                           /cmostest.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/NMOS
                                    /RESULTS/nmostest.co
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/TEST/NMOS
                                           /nmostest.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/TEST
                              /NMOS/RESULTS/nmostest.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/TEST
                                       /NMOS/nmostest.ci
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/TEST
                              /CMOS/RESULTS/cmostest.cif
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/TEST
                                       /CMOS/cmostest.ci
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cllmap.cmos
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/TEST
                                            /nmostest.co
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/TEST
                                           /cmostest.co
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/cmos_ts.cll
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rrplot.bin
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/cmostest.co
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/old_rplt.bin
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rrplot.bin.deb
/usr/local/cad/stanford/src/PLAGEN/PLA/TEST/test.pla
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/TEST/test.plg
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/TEST/RESULTS/test.pla
/usr/local/cad/stanford/src/aedcolors.h
/usr/local/cad/stanford/src/stipples.x
/usr/local/cad/stanford/celllib83
/usr/local/cad/stanford/celllib83/libs.cif
/usr/local/cad/stanford/lib/libacc.a
/usr/local/cad/stanford/lib/liblocal.a


DRAWER:  Statistics
UNIX DIRECTORY:  perm_stats
- TOOL LIST -


DRAWER:  Schedules
UNIX DIRECTORY:  perm_sched
- TOOL LIST -


DRAWER:  Design Data
UNIX DIRECTORY:  perm_data
- TOOL LIST -
```

DRAWER: Source Code
UNIX DIRECTORY: perm_sources
- TOOL LIST -

```
/usr/local/cad/stanford/src/DRC83/Makefile.bac
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/Makefile
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/Protomake.bak
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/Makefile.bak
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/Makefile
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/Makefile
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/Makefile
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/Makefile
/usr/local/cad/stanford/src/PLAGEN/PLA/Makefile
/usr/local/cad/stanford/src/PLAGEN/PLAGUE/Makefile
/usr/local/cad/stanford/lib/Makefile
/usr/local/cad/stanford/lib/tmake
/usr/local/cad/stanford/lib/Makefile.bak
/usr/local/cad/bin/nmake
/usr/local/cad/bin/makesm
/usr/local/cad/include
/usr/local/cad/include/plap
/usr/local/cad/include/plap/gconst.h
/usr/local/cad/include/plap/gtype.h
/usr/local/cad/include/plap/gvar.h
/usr/local/cad/include/plap/macros.h
/usr/local/cad/include/plap/primitive.h
/usr/local/cad/include/plap/prog.h
/usr/local/cad/include/plap/nodes.h
/usr/local/cad/include/plap/error.i
/usr/local/cad/include/plap/error.h
/usr/local/cad/uw/include
/usr/local/cad/uw/include/sys
/usr/local/cad/uw/include/sys/om_consts.h
/usr/local/cad/uw/include/sys/om_dmacros.h
/usr/local/cad/uw/include/sys/om_omacros.h
/usr/local/cad/uw/include/plap3
/usr/local/cad/uw/include/plap3/gconst.h
/usr/local/cad/uw/include/plap3/gtype.h
/usr/local/cad/uw/include/plap3/gvar.h
/usr/local/cad/uw/include/plap3/user.h
/usr/local/cad/uw/include/plap3/setup.h
/usr/local/cad/uw/include/plap3/attribute.h
/usr/local/cad/uw/include/plap3/bblocks.h
/usr/local/cad/uw/include/plap3/primitive.h
/usr/local/cad/uw/include/plap3/macros.h
/usr/local/cad/uw/include/plap3/nodes.h
/usr/local/cad/uw/include/plap3/error.h
/usr/local/cad/uw/include/plap3/nmos.h
/usr/local/cad/uw/include/plap3/isocmos.h
/usr/local/cad/uw/include/plap3/i2l.h
```

DRAWER: Source Code
UNIX DIRECTORY: perm_sources
- TOOL LIST - (Contd.)
/usr/local/cad/uw/include/plap3/i2ltype.h
/usr/local/cad/uw/include/plap3/cmospw.h
/usr/local/cad/uw/include/plap3/i2lext.h
/usr/local/cad/uw/include/plap3/nmosext.h
/usr/local/cad/uw/include/plap3/isocmosext.h
/usr/local/cad/uw/include/plap3/cmospwext.h
/usr/local/cad/uw/include/plap3/isocells.h
/usr/local/cad/uw/include/plap3/isopads.h
/usr/local/cad/lib/clay/PLA.c
/usr/local/cad/lib/clay/PLA_funcs.h
/usr/local/cad/lib/clay/README
/usr/local/cad/lib/clay/ext_def.h
/usr/local/cad/lib/clay/logic.c
/usr/local/cad/lib/clay/header.h
/usr/local/cad/lib/clay/new_def.h
/usr/local/cad/lib/clay/river.c
/usr/local/cad/lib/clay/clayread.h
/usr/local/cad/lib/clay/trans.c
/usr/local/cad/lib/clay/claywrite.h
/usr/local/cad/lib/clay/scn.h
/usr/local/cad/lib/clay/system_NMOS.h
/usr/local/cad/lib/clay/systemheader.h
/usr/local/cad/lib/clay/trace.h
/usr/local/cad/lib/clay/user_NMOS.h
/usr/local/cad/lib/clay/userheader.h
/usr/local/cad/stanford/src/DRC83/bool.c
/usr/local/cad/stanford/src/DRC83/boolf.c
/usr/local/cad/stanford/src/DRC83/bsort.c
/usr/local/cad/stanford/src/DRC83/btopen.c
/usr/local/cad/stanford/src/DRC83/butcondrop.c
/usr/local/cad/stanford/src/DRC83/butconexp.c
/usr/local/cad/stanford/src/DRC83/cmosextr.c
/usr/local/cad/stanford/src/DRC83/conv.c
/usr/local/cad/stanford/src/DRC83/cover.c
/usr/local/cad/stanford/src/DRC83/csort.c
/usr/local/cad/stanford/src/DRC83/diodechk.c
/usr/local/cad/stanford/src/DRC83/errout.c
/usr/local/cad/stanford/src/DRC83/exp.c
/usr/local/cad/stanford/src/DRC83/expand.c
/usr/local/cad/stanford/src/DRC83/frontsort.c
/usr/local/cad/stanford/src/DRC83/fullexp.c
/usr/local/cad/stanford/src/DRC83/impchk.c
/usr/local/cad/stanford/src/DRC83/intersect.c
/usr/local/cad/stanford/src/DRC83/isect_exc.c
/usr/local/cad/stanford/src/DRC83/mask.c
/usr/local/cad/stanford/src/DRC83/nmosextr.c
/usr/local/cad/stanford/src/DRC83/orcmos.c
/usr/local/cad/stanford/src/DRC83/ornmos.c
/usr/local/cad/stanford/src/DRC83/pnbuttcon.c
/usr/local/cad/stanford/src/DRC83/pplusdiff.c
/usr/local/cad/stanford/src/DRC83/pwellchk.c

DRAWER: Source Code
UNIX DIRECTORY: perm_sources
- TOOL LIST - (Contd.)

```
/usr/local/cad/stanford/src/DRC83/rectexp.c
/usr/local/cad/stanford/src/DRC83/rmbutcon.c
/usr/local/cad/stanford/src/DRC83/sep.c
/usr/local/cad/stanford/src/DRC83/separate.c
/usr/local/cad/stanford/src/DRC83/sepchk.c
/usr/local/cad/stanford/src/DRC83/sepdpbur.c
/usr/local/cad/stanford/src/DRC83/sepdpch.c
/usr/local/cad/stanford/src/DRC83/sepone.c
/usr/local/cad/stanford/src/DRC83/shrink.c
/usr/local/cad/stanford/src/DRC83/tranburied.c
/usr/local/cad/stanford/src/DRC83/trandiff.c
/usr/local/cad/stanford/src/DRC83/tranexp.c
/usr/local/cad/stanford/src/DRC83/trdiff.c
/usr/local/cad/stanford/src/DRC83/trpol.c
/usr/local/cad/stanford/src/DRC83/twocover.c
/usr/local/cad/stanford/src/DRC83/twoint_exc.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/add_depend.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2b.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2gram.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2gram.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2gram.y
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll_opt.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cstartact.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/dimension.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/intfunc.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/matrix.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/newcll.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/pwidth.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/s_ext.cll
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/set_ext.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/set_pass.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/switchtab.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/yymain.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cllmap.nmos
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cllmap.pmos
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/ncll2.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/nnew.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/Protomake.bak
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/errorfile
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/Makefile.bak
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/aadd_depend.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2b.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2gram.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2gram.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll2gram.y
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acll_opt.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/acstartact.c
```

DRAWER: Source Code
UNIX DIRECTORY: perm_sources
- TOOL LIST - (Contd.)

```
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/adimension.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/aintfunc.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/amatrix.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/anewcll.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/cll2_h.bak
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/apwidth.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/aset_ext.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/aset_pass.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CLL2/ayymain.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/concat.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/err2co.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/bane.1
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/lock.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/mktemp.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/window.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/xy2co.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/err2co.l
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/spathnames.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/Makefile
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/Protomake
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/bane.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/err2co.1
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/xy2co.1
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/READ_ME
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/lex.yy.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/spathnames.hba
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/aplot.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/plot.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/APLOT/plot.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/backupconcat.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/blockout.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/convert.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/merge.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rrplot.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/rsort.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/tplot.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/unconvert.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/window.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/font.GouldB
/usr/local/cad/stanford/src/PIPELINE83/SRC/RPLOT/font.h
/usr/local/cad/stanford/src/PLAGEN/PLA/pla.c
/usr/local/cad/stanford/src/PLAGEN/PLA/parse.h
/usr/local/cad/stanford/src/PLAGEN/PLA/pla.h
/usr/local/cad/stanford/lib/cifout-io.c
/usr/local/cad/stanford/lib/ckopen.c
/usr/local/cad/stanford/lib/cwd.c
/usr/local/cad/stanford/lib/data.c
/usr/local/cad/stanford/lib/dir.h
/usr/local/cad/stanford/lib/findargs.c
/usr/local/cad/stanford/lib/flsbuf.c
/usr/local/cad/stanford/lib/getaed.c
```

DRAWER: Source Code
UNIX DIRECTORY: perm_sources
- TOOL LIST - (Contd.)
/usr/local/cad/stanford/lib/malloc.c
/usr/local/cad/stanford/lib/pathtail.c
/usr/local/cad/stanford/lib/pen_plot.c
/usr/local/cad/stanford/lib/permalloc.c
/usr/local/cad/stanford/lib/scale.c
/usr/local/cad/stanford/lib/scanswitch.c
/usr/local/cad/stanford/lib/setext.c
/usr/local/cad/stanford/lib/stipples.c
/usr/local/cad/stanford/lib/stipples.h
/usr/local/cad/stanford/lib/swtab.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cif.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/user_act.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/yymain.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cif.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cifgram.y
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/scanswitch.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cifgram.h
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIF/cifgram.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifar.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifload.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/scifloader.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/splitfile.c
/usr/local/cad/stanford/src/PIPELINE83/SRC/CIFLOAD/cifload.h
/usr/local/cad/stanford/src/PLAGEN/PLA/parse.y
/usr/local/cad/bin/rrplot.c

DRAWER: Aux Data
UNIX DIRECTORY: perm_aux_data
- TOOL LIST -

/usr/local/cad/uw/lib
/usr/local/cad/uw/lib/libdb.a
/usr/local/cad/uw/lib/libzgp.a
/usr/local/cad/uw/lib/libom.a
/usr/local/cad/uw/lib/libplap3.a
/usr/local/cad/uw/lib/libplnmos.a
/usr/local/cad/uw/lib/libpli21.a
/usr/local/cad/uw/lib/libplisocmos.a
/usr/local/cad/uw/lib/libplcmospw.a
/usr/local/cad/uw/lib/libplisocell.a
/usr/local/cad/uw/lib/libplisopad.a
/usr/local/cad/uw/lib/psh
/usr/local/cad/uw/lib/psh/cells.psh
/usr/local/cad/uw/lib/psh/cmospw.psh
/usr/local/cad/uw/lib/psh/i21.psh
/usr/local/cad/uw/lib/psh/isocmos.psh
/usr/local/cad/uw/lib/psh/nmos.psh
/usr/local/cad/uw/lib/psh/pads.psh
/usr/local/cad/uw/lib/technology
/usr/local/cad/uw/lib/technology/nmos.tec

DRAWER: Aux Data
UNIX DIRECTORY: perm_aux_data
- TOOL LIST - (Contd.)
/usr/local/cad/uw/lib/technology/nmos.cmp
/usr/local/cad/uw/lib/technology/nmos.tec2
/usr/local/cad/uw/lib/technology/cmospw.tec
/usr/local/cad/uw/lib/technology/cmospw.cmp
/usr/local/cad/uw/lib/technology/cmospw.tec2
/usr/local/cad/uw/lib/technology/i2l.tec
/usr/local/cad/uw/lib/technology/isocmos.tec
/usr/local/cad/uw/lib/technology/isocmos.cmp
/usr/local/cad/uw/lib/technology/isocmos.tec2
/usr/local/cad/uw/bin/db2cif
/usr/local/cad/uw/bin/db7221
/usr/local/cad/uw/bin/db7580
/usr/local/cad/stanford/src/DRC83/mask.h
/usr/local/cad/stanford/src/DRC83/TEST/NMOS/Testburied.cll
/usr/local/cad/stanford/src/DRC83/TEST/NMOS/Testburied.co
/usr/local/cad/stanford/src/DRC83/TEST/NMOS/RESULTS
/usr/local/cad/stanford/src/DRC83/TEST/NMOS/RESULTS/Test.drc
/usr/local/cad/stanford/src/DRC83/TEST/NMOS/RESULTS/Testburied.drc
/usr/local/cad/stanford/src/DRC83/TEST/CMOS/cmos.cif
/usr/local/cad/stanford/src/DRC83/TEST/CMOS/cmos.cll
/usr/local/cad/stanford/src/DRC83/TEST/CMOS/cmos.co
/usr/local/cad/stanford/src/DRC83/TEST/CMOS/RESULTS/cmos.drc
/usr/local/cad/stanford/src/PIPELINE83/SRC/BANE/errorfile
/usr/local/cad/bin/nmos.tec
/usr/local/cad/bin/nmos.cmp
/usr/local/cad/bin/db7221

Appendix C: Source Code for a Partially Implemented VLSI CAD Toolkit Custodian

A prototype VLSI CAD toolkit Custodian has been designed to operate on the AFIT VAX 11/780. The custodian has been implemented using two programs: 1) A UNIX "C-Shell" script which contains the actual custodian code and 2) A program written in the "C" language designed to permit the author of the custodian "script" to specially condition the "script". The special conditioning is required so that the toolkit components may be accessed only via the custodian. Any attempts to read, write, or execute toolkit components without use of the custodian will be denied. The programs are presented in the two separate sections that follow.

Program 1: A "C-Shell" Custodian The program shown on the following pages will function as the "custodian" for the AFIT VLSI CAD Toolkit. This version is a prototype in the truest sense of the word and is provided to illustrate the structure and power of the toolkit custodian. To improve the program readability, the source code is printed in small type so that each statement may be displayed on a single line. Each line of the source code is numbered for easy reference. The reader is cautioned to delete the line numbers when transcribing the source to a format suitable for execution.

CUSTODIAN SOURCE CODE

```
#! /bin/csh
#    CUSTODIAN  Version 1.0  28 Nov 1984
#    Thomas R. Vermillion, Capt. USAF
#    AFIT/ENG Wright-Patterson AFB, Ohio
# reset the terminal
setenv TERMCAP /etc/termcap
tset >>& /dev/null
# set interrupt, suspend and quit to be the same as interrupt to prevent
# the user from interrupting the custodian
stty intr '^\' susp '^\' quit '^\'
#
# interruption of the custodian is not permitted yet
onintr -
# enable metacharacter filename expansions
unset  noglob
#
alias cd cd # just in case there is a strange alias in user environ
# set to ignore eof chars
set ignoreeof
#
clear
echo '        AFIT VLSI CAD TOOLKIT CUSTODIAN - Version 1.0 - 22 September 1984'
echo ' '
echo '        Unlocking VLSI CAD toolkit. There are 7 steps required.'
echo ' '
echo '- STEP - Remarks'
echo -n '-  01  - You are : '
#
#
# --------- BEGIN VARIABLE INITIALIZATION AREA -------------
# PATH TO TOP-LEVEL DIRECTORY FOR THE VLSI TOOLKIT
set TOOLKIT = /usr/local/cad/trv/vlsi_tool_kit
#
# FILE NAMES/PATHS
set tempath = $TOOLKIT/info_storage/temp_storage/temp_data/\$$
set data_access = $TOOLKIT/mgt_tools/data_access
set perm_area = $TOOLKIT/info_storage/perm_storage
set templog = $tempath".log"
set permlog = $data_access/'permanent.log'
set permdoc = $perm_area/perm_doc
set securlog = $TOOLKIT/mgt_tools/data_security/security.log
set toolkit_stats = $perm_area/perm_stats/usage_stats
set bakgnd_ok_list = $data_access/'bakgnd.list'
#
# set users to contain a list of the FILENAMES
#               which are of lists of users.
set users = `cat $data_access/userlists`
#
# GET USER IDENTIFICATION
set userid = `whoami`
echo $userid
```

```
#
echo -n '-  02  - Opening Toolkit Drawers : '
#
# TOOLKIT MANAGER IDENTIFICATION
set mgrid = 'hcarter'
set mgrname = 'Lt. Col H. Carter'
set mgraddr = 'Bldg. 640, Room 2xx'
set mgrphone = '255 - xxxx'
#
# STANDARD VARIABLES
set true = 1
set false = 0
set bell =  # ASCII BELL character
set main_menu = '0'
set done_main = $false
set done_skill = $false
set done_main_menu = $false
set access = 0
set skill = 0
set open_time = `date`
set printfil = ' '
set backgrnd = $false
# set the statistics
# the stat file has two entries per statistic, the first entry is
# the stat descriptor and the second is the stat value.  The first
# arguement in the stat file is a descriptor.
# the statistics variable structure is:
#
# stats[1] = "Current_Active_Toolkit_Sessions:"
# stats[2] = number of active sessions
# stats[3] = "Total_Toolkit_Sessions:"
# stats[4] =  total number of sessions
# stats[5] = "Total_Security_Violations;"
# stats[6] =  total number of security violations
set stats = `cat $toolkit_stats`
#
# TOOLKIT MANAGEMENT VARIABLES
#
# DUMMY VARIABLES
set dum_a = 0
set dum_b = 0
set dum_c = 0
set dum_d = 0
set dummy = ' '
#
set process = $$     # process id of this invocation
set toolselect = ' '
set flags = ' '
set infiles = ' '
set outfiles = ' '
set redirect = ' '
#
#----------- END VARIABLE INITIALIZATION AREA --------------
echo 'DONE.'
```

```
#
echo -n '-  03  - Session ID : '
set session_id = $open_time[3]$open_time[2]$process
echo $session_id
echo -n '-  04  - Creating Session Log : '
#
# open temporary log
echo ' ' >> $templog
echo '==========' >> $templog
echo -n 'SESSION OPENED AT : ' >> $templog
echo $open_time >> $templog
echo 'Session ID : '$session_id >>$templog
echo 'User ID : '$userid >> $templog
echo 'User Terminal Type : '$TERM >> $templog
echo 'User Directory : '$cwd >> $templog
echo ' ' >> $templog
echo 'Opening status of system storage : ' >> $templog
# save system status snapshot
w $userid >> $templog
df >> $templog
echo ' ' >> $templog
#
#
echo  'TOOLKIT STATISTICS: At session opening :' >> $templog
#loop through stat variables and save them
@ dum_a = 1
while ($dum_a <= $#stats)
@ dum_b = $dum_a + 1
echo $stats[$dum_a]' : '$stats[$dum_b] >> $templog
@ dum_a = $dum_a + 2
end  # while loop
# done with stat var save
echo ' ' >> $templog
echo 'DONE.'
echo -n '-  05  - Testing Your Directory : '
#
# Test to see if toolkit can access directory calling this
# program if no access is allowed then error message
#
if !( -r $cwd ) then
   set done_main = $true
endif # test if user can read his directory
#
if !( -w $cwd ) then
   set done_main = $true
endif # test if user can write to his directory
#
if !($done_main) then
   echo  'DIRECTORY ACCESS VERIFIED.'
else
   echo 'ACCESS DENIED : No access to CWD.' >> $templog
   set done_main = $true
   echo $bell'IMPROPER DIRECTORY ACCESS.'
   echo 'You must have READ & WRITE permission to Current Directory.'
```

```
    echo -n '    PRESS <RETURN> TO RETURN TO OPERATING SYSTEM -->'
    set dummy = ($<)
    echo -n '    PLEASE WAIT .. DO NOT DEPRESS ADDITIONAL KEYS.'
    goto close_kit
endif # CWD test
#
#
if !($done_main) then
# Establish User access Level
#
# Check each user LIST identified in the variable "users"
#
echo -n '-  06  - Toolkit Access is : '
@ access = 0
@ dum_a = 1
@ dum_b = $#users     # Number of user lists
while ($dum_a <= $dum_b)
  @ dum_c = `fgrep -x -c $userid $data_access/$users[$dum_a]`
  # if the user is in this list then give him access level of list
  if ($dum_c >= 1) then
    @ access = $dum_a
  endif
    @ dum_a = $dum_a + 1
end  # while loop
#
if !($access)  then
    echo 'ACCESS DENIED. : Not found in access lists.' >> $templog
    echo ' ' >> $securlog
    echo 'ILLEGAL ACCESS ATTEMPT : ' >> $securlog
    echo '                  User : '$userid >> $securlog
    echo '                  Time : '$open_time >> $securlog
    echo '            Session Id : '$session_id  >> $securlog
    echo ' ' >> $securlog
    set done_main = $true
    # security violation - update stats
    @ stats[6] = $stats[6] + 1
    echo $stats >! $toolkit_stats # update the stat file
    echo -n $bell
    echo $bell'DENIED.'
    echo     'You are not on the access list.'
    echo     'You must have pre-established access permission to use'
    echo     'the toolkit. To obtain toolkit access, you must do one'
    echo      of the following:'
    echo '      1:  Send a request via system mail to user : '$mgrid
    echo '      2:  Personally contact the following person:'
    echo '             '$mgrname
    echo '             '$mgraddr
    echo '             '$mgrphone
    echo ' '
    echo -n '     PRESS <RETURN> TO RETURN TO OPERATING SYSTEM -->'
    set dummy = ($<)
    echo -n '     PLEASE WAIT .. DO NOT DEPRESS ADDITIONAL KEYS.'
    goto close_kit
else
```

```
    # new session - update stats
    @ stats[2] = $stats[2] + 1  # number of active users
    @ stats[4] = $stats[4] + 1  # total number of sessions
    echo $stats >! $toolkit_stats # update the stat file
    echo  'GRANTED, level : '$access
    echo  'Toolkit access granted, level  : '$access >> $templog
    echo  '-  07  - Toolkit is OPEN.  Please record your session ID'
    echo  '         for use in the event you must refer to this session.'
    echo ' '
    echo -n $bell'        PRESS <RETURN> and wait for menu.'
    set dummy = ($<)
    echo -n '        PLEASE WAIT .. DO NOT DEPRESS ADDITIONAL KEYS.'
  endif #access  response messages
  endif # access test
  # NOW ALLOW USER INTERRUPTS .. BUT THEY JUMP TO GRACEFUL CLOSE
  onintr close_kit
  #
  # OBTAIN SKILL LEVEL
  set skill_menu = '0'
  while ( !($done_skill) && !($done_main) )
    clear
    echo '            ####### S K I L L  L E V E L #######'
    echo ' '
    echo '        1.  FIRST TIME User of Toolkit.'
    echo ' '
    echo '        2.  SOMEWHAT FAMILIAR with Toolkit.'
    echo ' '
    echo '        3.  VERY FAMILIAR with Toolkit.'
    echo ' '
    echo '        4.  EXPERT User.'
    echo ' '
    echo ' '
    echo -n '        Enter Your Choice and press <RETURN> -> '
    set  skill_menu = ($<)
  #
    echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
    switch ($skill_menu)

       case '1':
         echo 'Skill select : level '$skill_menu >> $templog
         set skill = $skill_menu
         set done_skill = $true
       breaksw

       case '2':
         echo 'Skill select : level '$skill_menu >> $templog
         set skill = $skill_menu
         set done_skill = $true
       breaksw

       case '3':
         echo 'Skill select : level '$skill_menu >> $templog
         set skill = $skill_menu
         set done_skill = $true
```

```
        breaksw

        case '4':
          echo 'Skill select : level '$skill_menu >> $templog
          set skill = $skill_menu
          set done_skill = $true
        breaksw

        default:
          echo 'Skill select : input error -> '$skill_menu >> $templog
          clear
          echo ' - Skill selection help being provided.' >> $templog
          more -d $peradoc/skill_help
           echo ' '
           echo -n '                              Press <RETURN> to continue -> '
           set dummy = ($<)
           echo ' '
           echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
        breaksw
    endsw # skill level
#
end # skill level while loop
#
#    M A I N   M E N U   L O O P
#
while ( !($done_main_menu) && !($done_main) )
  set main_menu = '0'
  clear
   echo '              ########   M A I N   M E N U   ########'
   echo ' '
   echo '        1.  EXIT and return to Unix.'
   echo ' '
   echo '        2.  EXECUTE a toolkit component.'
   echo ' '
   echo '        3.  DISPLAY your directory contents.'
   echo ' '
   echo '        4.  CHANGE your working directory.'
   echo ' '
   echo '        5.  VIEW or PRINT a file.'
   echo ' '
   echo '        6.  Get HELP.'
#
# user access to these menu options depends upon access level
#
  if ($access > 1) then
     echo ' '
     echo '        7.  Generate REPORTS from the toolkit.'
  endif # access level > 1 options

  if ($access >= 2) then
     echo ' '
     echo '        8.  Execute a toolkit UTILITY program.'
  endif # access >= 2

```

```
    if ($access >= 3) then
       echo ' '
       echo '         9.  MODIFY the toolkit.'
    endif # access level 3 option
#
    echo ' '
    echo ' '
    echo -n '         Enter Your Choice and press <RETURN> -> '
    set  main_menu = ($<)
#
    echo ' '
    echo -n $bell'                  PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'

#
    switch ($main_menu)

       case '1':
         echo 'Main select : EXIT to Unix.' >> $templog
         set done_main = $true
         goto close_kit
       breaksw

       case '2':
          clear
          echo  'Main select : EXECUTE tool described below: ' >> $templog
          set command
          set component
          set infiles
          set outfiles
          set flags
          # for highly skilled users
          if ($skill >= 3) then
            echo 'Enter the complete command line for the component :'
            set component = ($<)
          # for the less experienced users
          else
            echo -n 'Enter the name of the tool to execute -> '
               set component = ($<)
            echo -n 'Enter the names of all input files -> '
               set infiles = ($<)
            echo -n 'Enter the name of the output file -> '
               set outfiles = ($<)
            echo -n 'Enter the command flags -> '
               set flags = ($<)
          endif # skill >= 3
          echo ' '
          echo -n $bell'                  PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
          clear
          # have complete input now in the variable - command
          set command = "$component $flags $infiles $outfiles"
          echo $command >> $templog
          # test to see if allowed to execute this component
          # authorized lists maintained as "toolsX" where the X
          # represents the user access level
```

```
            set dum_a = `fgrep -x -c $component[1] $data_access/tools$access`
              if ($dum_a) then
                echo ' - Execution Authorized.' >> $templog
                # now see if tool exists
                #
                set dum_b = `find $TOOLKIT -name $component[1] -print`
                echo ' - Tool location is: ' >> $templog
                echo '  - '$dum_b >> $templog
                if ( -e  $dum_b ) then
                  echo ' '
                  echo -n 'EXECUTE in the F(oreground) or B(ackground) ? ->'
                  set dummy = ($<)
                  echo ' '
                  echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'

                  switch ($dummy)
                     case 'F':
                       clear
                       set backgrnd = $false
                       echo  ' - Executing in foreground.' >> $templog
                       echo  'Executing - '$component[1]
                       $command
                       echo ' - Execution completed, status = '$status >> $templog
                       echo 'Execution completed, status = '$status
                      breaksw # foreground

                     case 'B':
                       clear
                       set dum_a = `fgrep -x -c $dum_b $bakgnd_ok_list`
                       if ($dum_a) then
                         echo ' - Background Execution Authorized.' >> $templog
                         set backgrnd = $true
                         echo  ' - Executing in background.' >> $templog
                         echo  -n 'Executing in background : as Process Number ->'
                         $command& # execute the job
                       else
                         echo ' - Background Execution NOT Authorized.' >> $templog
                         echo  ' - EXECUTION ABORTED.' >> $templog
                      echo  $bell'EXECUTION ABORTED. - Background Execution NOT Authorized.'
                         echo ' '
                         echo -n '                         Press <RETURN> to continue -> '
                         set dummy = ($<)
                         echo ' '
                         echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
                       endif # see if background execution is ok
                      breaksw # background

                      default:
                       clear
                       echo  ' - EXECUTION ABORTED. No User Selection.' >> $templog
                      breaksw # background default

                   endsw # foreground or background
                else # tool cannot be executed
```

```
                clear
                echo 'Can not execute - '$component[1] >> $templog
                echo 'Can not execute - '$component[1]
              endif # -e dum_b existence test
            else # user is not authorized to execute this tool
              clear
              echo ' - Not an authorized execution - '$component[1] >> $templog
              echo 'Not an authorized execution - '$component[1]
            endif # dum_a execution test
          echo ' '
          echo -n '                          Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
       breaksw

       case '3':
          echo 'Main select : DISPLAY directory.' >> $templog
          clear
          $TOOLKIT/design_tools/utilities/dir -la | more -d # prints directory
          echo ' '
          echo -n '                          Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
       breaksw

       case '4':
          # NOTE CANNOT USE SOME "WILDCARD" SUBSTITUTIONS HERE
          echo 'Main select : CHANGE directory to : ' >> $templog
          clear
          echo -n 'Enter NEW directory name ->'
          set dummy = ($<)
          echo ' '
          echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
          echo ' - '$dummy >> $templog
          if ($#dummy == '0') then
             set dummy = 'pwd'
          endif # safety factor for only a return

          if ( $dummy == ~ ) then
             set dummy = $HOME
          endif # safety factor for only a tilde

          if ($dummy == .) then
             set dummy = 'pwd'
          endif # safety factor for only a period

          if !( -r $dummy )  then
            clear
            echo ' - DIRECTORY CHANGE IGNORED : Cannot change.' >> $templog
            echo ' '
            echo $bell'CHANGE IGNORED.  Cannot change to selected directory.'
          else
```

```
            cd $dummy
            echo ' '
            echo  'Directory CHANGED to : '$dummy
            echo  ' - Directory changed.' >> $templog
          endif # read authorization test for new dir
          echo ' '
          echo -n '                          Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
       breaksw

       case '5':
              clear
              echo 'Main select : VIEW or PRINT files' >> $templog
              echo -n 'Do you wish to V(iew) or P(rint) files ? ->'
              set dummy = ($<)
              echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
              switch ($dummy)

                case 'P':  # UPPER CASE REQUIRED FOR ACCURACY
                 echo 'Main select :  PRINT file listed below:' >> $templog
                 clear
                 echo    'Enter the exact name(s) of the file(s) you wish to print and press <RETURN>. '
                 echo    ' '
                 set prntfile = ($<)
                 echo ' '
                 echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
                 clear
                 if ($#prntfile == '0') then # invalid request
                   echo 'You should enter a filename.'$bell
                   echo ' - NO filename arguements.' >> $templog
                 else
                   echo '  - '$prntfile >> $templog
                   # test to see if user has read access before printing
                   foreach dum_a ($prntfile)
                   if ( -r $dum_a ) then
                     # print only ordinary files
                      if ( -f  $dum_a ) then
                       echo  ' - Printing.' >> $templog
                       echo  'Printing - '
                       echo -n ' - '$dum_a
                       vpr $dum_a
                       echo ' - DONE PRINTING.'$bell
                       echo ' - DONE Printing.' >> $templog
                      else
                       echo ' - PRINT ABORTED: File is not printable.' >> $templog
                       echo ' - PRINT ABORTED: File is not printable.'$bell
                      endif # printable test for file
                    else
                      echo 'PRINT ABORTED: You may not read the specified file.'$bell
                      echo ' - PRINT ABORTED: User does not have read access to specified file.' >> $templog
                    endif # read access test
                    end #foreach file name entered
```

```
      endif # number of filenames entered
      echo ' '
      echo -n '                        Press <RETURN> to continue -> '
      set dummy = ($<)
      echo ' '
      echo -n $bell'               PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
     breaksw # print

     case 'V':  # UPPER CASE REQUIRED FOR ACCURACY
      echo 'Main select : VIEW files listed below:' >> $templog
      clear
      echo    'Enter the exact name(s) of the file(s) you wish to view and press <RETURN>. '
      echo    ' '
      set viewfile = ($<)
      echo ' '
      echo -n $bell'               PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
      clear
      if ( $#viewfile == '0' ) then # no arguments
        echo 'You should enter a filename.'$bell
        echo ' - NO filename arguments.' >> $templog
      else
        echo '  - '$viewfile >> $templog
        # test to see if user allowed to view the files
        foreach dum_a ($viewfile)
        if ( -r $dum_a ) then
          # view only ordinary files
          if ( -f  $dum_a ) then
           echo  '  - Viewing. ' >> $templog
           echo  'VIEWING : '$dum_a
           more -d $dum_a
           echo '  - DONE viewing.' >> $templog
          else
           clear
           echo '  - VIEW ABORTED: '$dum_a' is not ASCII.' >> $templog
           echo $bell'VIEW ABORTED: '$dum_a' is not ASCII.'
          endif # printable test for file
         else
          clear
          echo $bell'VIEW ABORTED: You may not view the specified file.'$bell
          echo '  - VIEW ABORTED: User may not view the specified file.' >> $templog
         endif # read access test
         echo ' '
         echo -n '                        Press <RETURN> to continue -> '
         set dummy = ($<)
         echo ' '
         echo -n $bell'               PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
         end # foreach dum_a in dummy
      endif # number of filenames entered
     breaksw # view

     default:
      echo 'VIEW or PRINT : Invalid option. You must use UPPER CASE.'
      echo 'VIEW or PRINT : Invalid option -> '$dummy >> $templog
      echo ' '
```

```
                echo -n '                       Press <RETURN> to continue -> '
                set dummy = ($<)
                echo ' '
                echo -n $bell'                  PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
               breaksw # default view or print
             endsw # view or print
      breaksw # main case 5

      case '6':
         echo 'Main Select : HELP' >> $templog
         set donelearn = $false
         # HELP Menu
         while !($donelearn)
         clear
          set infoselect = '0'
          echo '              #### I N S T R U C T I O N S ####'
          echo ' '
          echo '        1.  EXIT the HELP Mode.'
          echo ' '
          echo '        2.  Search UNIX Manuals by KEYWORD.'
          echo ' '
          echo '        3.  Read a particular UNIX manual entry.'
          echo ' '
          echo '        4.  Read help information for a particular COURSE.'
          echo ' '
          echo '        5.  Read HELP information for a particular TOOL.'
          echo ' '
           if ($access > 1) then
             echo '        6.  Read HELP for toolkit REPORTS.'
             echo ' '
           endif # access > 1
           if ($access > 2) then
             echo '        7.  Read HELP for toolkit MODIFICATIONS.'
             echo ' '
             echo '        8.  Read HELP for toolkit UTILITY programs.'
             echo ' '
           endif # access > 2

          echo -n '        Enter Your Choice then press <RETURN> -> '
          set infoselect = ($<)
          echo ' '
          echo -n $bell'                  PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'

        switch ( $infoselect )

          case '1':
             echo 'Help Select : EXIT HELP' >> $templog
             set donelearn = $true
          breaksw

          case '2':
             echo -n 'Help Select : Unix Key Search using - ' >> $templog
             clear
             echo -n 'Enter Keyword(s) -> '
```

```
          set dummy = ($<)
          echo $dummy >> $templog
          clear
          echo 'Searching UNIX manuals ... please wait.'
          echo ' '
          man -k $dummy | more
          echo ' '
          echo -n '                        Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                   PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
       breaksw

       case '3':
          echo -n 'Help Select : Read UNIX manual - ' >> $templog
          clear
          echo -n 'Enter UNIX Manual ID -> '
          set dummy = ($<)
          echo $dummy >> $templog
          clear
          echo 'Locating UNIX manual -> '$dummy' ... please wait.'
          echo ' '
          man $dummy
          echo ' '
          echo -n '                        Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                   PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
       breaksw

       case '4':
          echo -n 'Help Select : Course - ' >> $templog
          clear
          echo -n 'Enter Course ID -> '
          set dummy = ($<)
          echo $dummy >> $templog
          clear
          echo 'Searching for HELP for course -> '$dummy' ... please wait.'
          echo ' '
          if ( -e $peradoc/$dummy'_help' ) then
            clear
            echo ' - HELP for course : '$dummy' presented.' >> $templog
            echo 'HELP for course -> '$dummy
            echo ' '
            more -d $peradoc/$dummy'_help'
          else
            echo 'Help for course '$dummy' is not available.'
            echo ' - Help for course '$dummy' not available.' >> $templog
          endif # look for the help file
          echo ' '
          echo -n '                        Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                   PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
```

```
          breaksw

          case '5':
            echo -n 'Help Select : Tool - ' >> $templog
            clear
            echo -n 'Enter Tool Name -> '
            set dummy = ($<)
            echo $dummy >> $templog
            clear
            echo 'Searching for HELP for tool - '$dummy' ... please wait.'
            echo ' '
            if ( -e $permdoc/$dummy'_help' ) then
              clear
              echo ' - HELP for tool : '$dummy' presented.' >> $templog
              echo 'HELP for tool -> '$dummy
              echo ' '
              more -d $permdoc/$dummy'_help'
            else
              echo $bell'Help for tool '$dummy' is not available.'
              echo ' - Help for tool '$dummy' not available.' >> $templog
            endif # look for the help file
            echo ' '
            echo -n '                    Press <RETURN> to continue -> '
            set dummy = ($<)
            echo ' '
            echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
          breaksw

          case '6':
           if ($access >1) then
            echo 'Help Select : toolkit REPORTS ' >> $templog
            clear
            echo 'Locating HELP for toolkit REPORTS ... please wait.'
            echo ' '
            if ( -e $permdoc/'reports_help' ) then
              clear
              echo ' - HELP for REPORTS  presented.' >> $templog
              echo 'HELP for REPORTS : '
              echo ' '
              more -d $permdoc/'reports_help'
            else
              echo $bell'Help for REPORTS is not available.'
              echo ' - Help for REPORTS not available.' >> $templog
            endif # look for the help file
            echo ' '
            echo -n '                    Press <RETURN> to continue -> '
            set dummy = ($<)
            echo ' '
            echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
           endif # access > 1
          breaksw

          case '7':
           if ($access > 2) then
```

```
             echo 'Help Select : toolkit MODIFICATIONS' >> $templog
             clear
             echo 'Locating HELP for toolkit MODIFICATIONS ... please wait.'
             echo ' '
             if ( -e $permdoc/'mods_help' ) then
               clear
               echo ' - HELP for MODIFICATIONS presented.' >> $templog
               echo 'HELP for MODIFICATIONS : '
               echo ' '
               more -d $permdoc/'mods_help'
             else
               echo $bell'Help for MODIFICATIONS is not available.'
               echo ' - Help for MODIFICATIONS not available.' >> $templog
             endif # look for the help file
             echo ' '
             echo -n '                          Press <RETURN> to continue -> '
             set dummy = ($<)
             echo ' '
             echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
            endif # access > 2
          breaksw

          case '8':
            if ($access > 2) then
             echo 'Help Select : toolkit UTILITIES' >> $templog
             clear
             echo 'Locating HELP for toolkit UTILITIES ... please wait.'
             echo ' '
             if ( -e $permdoc/'utils_help' ) then
               clear
               echo ' - HELP for UTILITIES presented.' >> $templog
               echo 'HELP for UTILITIES : '
               echo ' '
               more -d $permdoc/'utils_help'
             else
               echo $bell'Help for UTILITIES is not available.'
               echo ' - Help for UTILITIES not available.' >> $templog
             endif # look for the help file
             echo ' '
             echo -n '                          Press <RETURN> to continue -> '
             set dummy = ($<)
             echo ' '
             echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
            endif # access > 2
          breaksw
        endsw # help menu switch
      end # help menu while loop
      breaksw # main menu switch for HELP
#
# back to main menu case choices
#
      case '7':
         set done_report= $false
         if ($access > 1) then
```

```
while !($done_report)
 clear
 echo 'Main select: - Generate REPORTS.' >>$templog
# REPORTS MENU
    set reptselect = '0'
    echo '         * * * *  T O O L K I T   R E P O R T   M E N U   * * * *'
    echo ' '
    echo ' '
    echo '         1.  USER ACTIVITY report for specific users.'
    echo ' '
    echo '         2.  SESSION count for particular month.'
    echo ' '
    echo '         3.  EXIT report menu.'
    echo ' '
    echo -n '          Select ONE option then press <RETURN> -> '
    set reptselect = ($<)
    echo -n $bell'                 PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
    switch ($reptselect)

      case '1':
        echo 'Report select : USER ACTIVITY' >> $templog
        clear
        echo 'Enter the complete path to filename with target users.'
        set dum_a
        set dum_a = ($<)
        echo ' '
        echo -n 'Please wait ... reading your list - '
        if ($#dum_a > 0) then
        if ( -e $dum_a) then
          set dum_b =  `cat $dum_a`
          echo 'DONE.'
          # open a temp file to hold report
          echo '   ******  USER ACTIVITY REPORT  ******' >> $tempath".act"
          echo ' ' >> $tempath".act"
          echo -n '     As of ' >> $tempath".act"
          date >> $tempath".act"
          echo ' ' >> $tempath".act"
          echo 'User ID   Activity' >> $tempath".act"
          echo ' ' >> $tempath".act"
          # send the report to user terminal
          clear
          echo '   ******  USER ACTIVITY REPORT  ******'
          echo ' '
          echo -n '     As of '
          date
          echo ' '
          echo 'User ID   Activity'
          echo ' '
          foreach dum_c ($dum_b)
          set dum_d = `fgrep -x -c 'User ID : '$dum_c $permlog`
          echo $dum_c'      '$dum_d >> $tempath".act"
          echo $dum_c'      '$dum_d
          end # foreach
          echo ' '
```

```
                echo -n 'PLEASE WAIT - printing report -'
                vpr $tempath".act"
                echo $bell'DONE.'
                echo ' - Report completed.' >> $templog
              else
               echo $bell'Cannot open your user list.'
               echo ' - Cannot open user list.'>> $templog
              endif # read user list
              else # not enough arguments
                echo $bell'NO FILE NAME of users to examine.'
                echo ' - NO FILE NAME of users to examine.' >> $templog
              endif #arguments test
             echo ' '
             echo -n '                              Press <RETURN> to continue -> '
             set dummy = ($<)
             echo ' '
             echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
            breaksw

            case '2':
              echo - 'Report select : SESSION COUNT for - ' >> $templog
              clear
              echo -n 'Enter the three-letter abbreviation for month -> '
              set dum_a = ($<)
              echo ' '
              echo $dum_a >> $templog
              set dum_b = `egrep -c 'Session ID : ..'$dum_a $permlog`
              echo ' '
              echo 'There were '$dum_b' sessions in '$dum_a >> $templog
              echo 'There were '$dum_b' sessions in '$dum_a
              echo ' '
              echo -n '                              Press <RETURN> to continue -> '
              set dummy = ($<)
              echo ' '
              echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
            breaksw

            case '3':
               set done_report = $true
               echo 'Report select: - EXIT.' >> $templog
            breaksw # exit report menu

            default:
              echo 'Report select : INVALID selection.' >> $templog
            breaksw
          endsw # report menu
       end # report while loop
       endif # access > 1
    breaksw

    case '8':
       # UTILITIES MENU
       if ($access > 2) then
          clear
```

```
set utilselect = '0'
echo '        * * * *  U T I L I T Y   P R O G R A M   M E N U   * * * *'
echo ' '
echo ' '
echo '        1.  ADD to An Authorized User List'
echo ' '
echo '        2.  DELETE from An Authorized User List'
echo ' '
echo '        3.  VIEW Toolkit STATISTICS'
echo ' '
echo ' '
echo -n '         Select ONE option then press <RETURN> -> '
set utilselect = ($<)
echo -n $bell'                 PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
switch ($utilselect)
 case '1':
   echo 'Util Select : ADD to User List' >> $templog
   clear
   echo -n 'Additions filename  -> '
   set dummy = ($<)
   echo ' - Additions list is : '$dummy >> $templog
   echo 'Please Wait. Testing List.'
   if ( -r $dummy) then
     set dum_a = `cat $dummy`
     echo -n 'List Read. Add these users to access level (1-3) -> '
     set dummy = ($<)
     echo ' - Add to user list for level '$dummy >> $templog
     clear
     echo 'PROCESSING ADDITIONS : Please Wait.'
     echo
     foreach dum_b ($dum_a)
       set dum_c = `fgrep -x -c $dum_b $data_access/$users[$dummy]`
       if ($dum_c < 1) then
          echo $dum_b >> $data_access/$users[$dummy]
          echo $dum_b' - added to '$users[$dummy]
          echo $dum_b' added to '$users[$dummy] >> $templog
       else
          echo $dum_b' - ignored ... already in list.'
       endif # dum_c test
     end # foreach dum_b
     echo ' '
     echo 'Additions Completed.'$bell
     echo ' - Additions Completed.' >> $templog
   else
     echo 'ADD ABORTED: Could not read specified file.'$bell
     echo 'ADD ABORTED: Could not read specified file.' >> $templog
   endif # test for read access
   echo ' '
   echo -n '                      Press <RETURN> to continue -> '
   set dummy = ($<)
   echo ' '
   echo -n $bell'                 PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
  breaksw

```

```
            case '2':
              echo 'Util Select : DELETE from User List' >> $templog
              clear
              echo -n 'Deletions filename  -> '
              set dummy = ($<)
              echo 'Please Wait.  Testing list.'
              echo ' - Deletion list is: '$dummy >> $templog
              if ( -r $dummy) then
                set dum_a = `cat $dummy`
                echo -n 'List Read. What access level (1-3) are these users -> '
                set dummy =  ($<)
                echo ' - Delete from user list for level '$dummy >> $templog
                clear
                echo 'PROCESSING DELETIONS. Please Wait.'
                echo ' '
                set dum_b = `cat $data_access/$users[$dummy]`
                echo -n ' ' >> $tempath".lst" # create a temp list
                foreach dum_c ($dum_a)
                  foreach dum_d ($dum_b)
                    if ($dum_d == $dum_c) then
                      echo ' - '$dum_c' deleted.' >> $templog
                      echo $dum_c' - deleted.'
                    else
                      echo $dum_d >> $tempath".lst"
                    endif # dum_d test
                  end # foreach dum_d
                  set dum_b = `cat $tempath".lst"`
                  rm -f $tempath".lst"
                end # foreach dum_c
                rm -f $data_access/$users[$dummy]
                echo $dum_b > $data_access/$users[$dummy]
                echo 'Deletions Completed.'
                echo ' - Deletions Completed.' >> $templog
              else
                echo 'DELETE ABORTED: Could not read specified file.'$bell
                echo 'DELETE ABORTED: Could not read specified file.'>> $templog
              endif # read access test
              echo ' '
              echo -n '                         Press <RETURN> to continue -> '
              set dummy = ($<)
              echo ' '
              echo -n $bell'                 PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
            breaksw

            case '3':
             echo 'Util Select: VIEW Statistics' >> $templog
             clear
             echo  -n 'TOOLKIT STATISTICS: As of '
             date
             echo ' ' >> $templog
             echo  -n 'TOOLKIT STATISTICS: As of ' >> $templog
             date >> $templog
             echo ' '
             @ dum_a = 1
```

```
                while ($dum_a <= $#stats)
                @ dum_b = $dum_a + 1
                echo $stats[$dum_a]' : '$stats[$dum_b]
                echo $stats[$dum_a]' : '$stats[$dum_b] >> $templog
                @ dum_a = $dum_a + 2
                end  # while loop
                echo ' '
                echo -n '                          Press <RETURN> to continue -> '
                set dummy = ($<)
                echo ' '
                echo -n $bell'                 PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
               breaksw

               default:
                 echo ' - No util menu selection.' >> $templog
               breaksw # default util menu


           endsw # utility menu switch
          endif # access >= 2
        breaksw # main case 8



       case '9':
          # MODIFICATION MENU
          if ($access > 2) then
             clear
             set modselect = '0'
             echo  '        * * * *   M O D I F I C A T I O N    M E N U    * * * *'
             echo ' '
             echo ' '
             echo '         1.  ADD a toolkit component.'
             echo ' '
             echo '         2.  DELETE a toolkit component.'
             echo ' '
             echo '         3.  MOVE a toolkit component.'
             echo ' '
             echo ' '
             echo -n '        Select ONE option then press <RETURN> -> '
             set modselect = ($<)
             echo -n $bell'                  PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
             switch ($modselect)

             case '1':
              clear
              echo 'Mods select: ADD component.' >> $templog
              echo 'Enter complete current path to component:'
              set dum_a = ($<)
              echo 'Enter complete destination path for component:'
              set dum_b =($<)
              if ( -r $dum_a ) then
                 echo 'Adding new component:'
                 echo '  From: '$dum_a
```

```
          echo '    To: '$dum_b
          echo 'Adding new component:' >> $templog
          echo '  From: '$dum_a >> $templog
          echo '    To: '$dum_b >> $templog
          cp $dum_a $dum_b
          echo $bell'  COMPONENT ADDED.'
       endif # -r read test
       echo ' '
       echo -n '                         Press <RETURN> to continue -> '
       set dummy = ($<)
       echo ' '
       echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
      breaksw

      case '2':
       clear
       echo 'Mods select: DELETE component.' >> $templog
       echo 'Enter complete current path to component:'
       set dum_a = ($<)
       if ( -w $dum_a ) then
          echo 'Deleting component: '$dum_a
          echo 'Deleting component: '$dum_a >> $templog
          echo -n 'ARE YOU SURE ? Y(es) N(o) -> '
          set dummy = ($<)
          switch ($dummy)

           case 'Y':
               echo 'Deletion verified.' >> $templog
               rm -f $dum_a
               echo $dum_a' removed.'
               echo $dum_a' removed.' >> $templog
            breaksw

           default:
               echo 'CAPITAL Y not entered.'
               echo 'Deletion ABORTED at user request.'
               echo 'Deletion ABORTED at user request.' >> $templog
           breaksw
          endsw # dummy
       else
         echo 'Deletion ABORTED : Could not delete.' >> $templog
         echo 'Deletion ABORTED : Could not delete.'
       endif # -w write test
       echo ' '
       echo -n '                         Press <RETURN> to continue -> '
       set dummy = ($<)
       echo ' '
       echo -n $bell'                    PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
      breaksw

      case '3':
       clear
       echo 'Mods select: MOVE component.' >> $templog
       echo 'Enter complete current path to component:'
```

```
                set dum_a = ($<)
                echo 'Enter complete destination path for component:'
                set dum_b =($<)
                if ( -r $dum_a ) then
                   echo 'Moving component:'
                   echo '  From: '$dum_a
                   echo '    To: '$dum_b
                   echo 'Moving component:' >> $templog
                   echo '  From: '$dum_a >> $templog
                   echo '    To: '$dum_b >> $templog
                   mv $dum_a $dum_b
                   echo $bell'MOVE COMPLETED.'
                else
                   echo 'Move ABORTED : Cannot read source.' >> $templog
                   echo 'Move ABORTED : Cannot read source.' >> $templog
                endif # -r read test
                echo ' '
                echo -n '                        Press <RETURN> to continue -> '
                set dummy = ($<)
                echo ' '
                echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
               breaksw

               default:
                  echo '  - No mod menu selection.' >> $templog
               breaksw # default mods

      endsw #  mod menu
          endif # access > 2
      breaksw # main menu case 9

        default:
          echo 'User being provided HELP with MAIN select.' >> $templog
          clear
          more -d $permdoc/main_help
          echo ' '
          echo -n '                        Press <RETURN> to continue -> '
          set dummy = ($<)
          echo ' '
          echo -n $bell'                PLEASE WAIT. DO NOT DEPRESS ADDITIONAL KEYS.'
        breaksw

     endsw # main menu switch

   end # main menu while loop
   #
   #  E N D   O F   M A I N   S E L E C T I O N   L O O P
   #
   close_kit: # label for jump after user interrupt of program
   # **** NEXT COMMAND REMOVES INTERRUPT CAPABILITY TO SHELL
   #      UNTIL THE TOOLKIT IS CLOSED
   onintr -
   #
   clear
```

```
if ($main_menu != '1' && $done_main != '1') then
   echo 'I N T E R R U P T E D - BY USER ' >> $templog
   echo $bell'                        I N T E R R U P T E D - BY USER'
   echo ' '
endif # report interrupt
echo '        Locking VLSI CAD toolkit. There are 5 steps required.'
echo ' '
echo '- STEP - Remarks'
echo -n '-  01  - Saving System Status : '
echo ' ' >> $templog
echo -n 'SESSION CLOSED AT : ' >>$templog
date >> $templog
w -u >> $templog
echo ' ' >> $templog
echo 'Closing status of system storage : ' >> $templog
df >> $templog
echo 'SAVED.'
echo -n '-  02  - Updating Statistics : '
if ($access) then
  @ stats[2] =  $stats[2] - 1   # number of active users reduced by 1
endif # delete one user if there was a valid user session
echo $stats > $toolkit_stats # update the stat file
echo 'UPDATED.'
echo -n '-  03  - Closing Session Log : '
echo ' ' >> $templog
echo  'TOOLKIT STATISTICS: At session close:' >> $templog
@ dum_a = 1
while ($dum_a <= $#stats)
@ dum_b = $dum_a + 1
echo $stats[$dum_a]' : '$stats[$dum_b] >> $templog
@ dum_a = $dum_a + 2
end  # while loop
echo ' ' >> $templog
if ($backgrnd) then
  set dum_a = `ps`
  echo $dum_a >> $ templog
  echo '' >> $templog
endif # save process status to see what jobs were running
echo '==========' >> $templog
#
cat $templog >> $permlog
echo 'DONE.' # closing the logs
# remove all temporary files created during this session
echo -n '-  04  - Removing Temporary Files : '
#
rm -f $tempath*
echo 'DONE.'
echo -n '-  05  - Returning control to the operating system ...'
# end session - update stats
stty intr '^?' susp '^Z' quit '^\'
clear
if ($backgrnd) then
  echo 'You initiated background jobs during this session'
  echo ' '
```

```
    echo 'Here is the status of your current processes:'
    echo ' '
    echo $dum_a
    echo ' '
  endif # print process status just in case job is still running
  #
  #
  # END OF CUSTODIAN "C -SHELL " PROGRAM
```

Program 2: A "C" Program Used to Protect The Toolkit

The program listed below, written in the "C" language, has been provided to this author courtesy of Mr Joe Hamlin, AFIT/AD. The manager of the toolkit may effectively "hide" the toolkit from direct access by other users by performing two tasks: 1) giving read, write, and execute permission for all toolkit components only to the "owner" of the custodian script and 2) conditioning the custodian script with the program provided below. The program, setu.c, conditions a script by setting the script's user permissions so that the script will respond to a user as if the user were its "owner". In this way, users may access toolkit components via the custodian and its "conditioned" permissions even though the users do not possess operating system permissions to directly access toolkit components.

```
/***************** BEGIN setu.c ************/

main(argc, argv, envp)
int argc;
char *argv[], **envp;
{
char *strcat();
char cmd[100];

cmd[0]='\0';
strcat(cmd, "/usr/local/cad/trv/vlsi_tool_kit/scr.");
strcat(cmd, argv[0]);
setgid(31);
execve(cmd, argv, envp);
}

/*****************END setu.c ***************/
```

The implementation of this protection program is accomplished as shown below:

1. Edit setu.c so that it contains the correct path name to your script. Like: /usr/local/cad/trv/vlsi_cad_toolkit/scr.custodian.

2. The script should have a name like
   scr.name

3. The first line in the script must be
   #!/bin/csh

4. Compile setu.c with
   cc -o setu setu.c

5. Set special protection bits with
   chmod 4711 setu

6. Move to the directory where the script is to be used.

7. Type: "ln setu name". This links "setu" with the script "name". When the script "name" is executed, it will execute as though the user had ownership permissions.

Appendix D: Application of the Evaluation Metrics to the Prototype Custodian

AFIT VLSI CAD Toolkit Custodian Evaluation Formulas

The evaluation formulas and scales provided below, copied from chapter 5, have been used to evaluate the prototype custodian detailed in Appendix C.

Drawer Component Evaluation Formula:

$$DCQ = \frac{\sum_{1}^{i} [\ EV(i) \times CL(i)\ ]}{n} \qquad \text{(EQ 1)}$$

Where :

DCQ = *Drawer Component* Quality (overall)
EV = Essentiality Value of characteristic i
CL = Characteristic Level of characteristic i
n = Number of characteristics being evaluated

Drawer Evaluation Formula:

$$DQ = \frac{\sum_{1}^{i} [\ EV(i) \times DCQ(i)\ ]}{n} \qquad \text{(EQ 2)}$$

Where :

DQ = Drawer Quality (overall)
DCQ = Drawer Component Quality of drawer component i
EV = Essentiality Value of drawer component i
n = Number of drawer components

Toolkit Evaluation Formula:

$$TQ = \frac{\sum_{1}^{i} [\ EV(i) \ \times \ DQ(i)\ ]}{n} \qquad \text{(EQ 3)}$$

Where :

TQ = Toolkit Quality (overall)
DQ = Drawer Quality of drawer i
EV = Essentiality Value of drawer i
n = Number of drawers

Evaluation Scales:

The scale shown below provides a measure of the essentiality of a characteristic to the success of toolkit integration.

Essentiality Value (EV)	Essentiality of Characteristic To Successful Integration
32	Extremely important to success (Certain failure if missing)
16	Important to success (Probable failure if missing)
8	Fairly important to success (Possible failure if missing)
2	Has some impact upon success (No predictable impact upon success if missing but less than optimum integration is probable)
1	Slight impact upon success (No predictable impact upon success if missing but less than optimum integration is certain)

The scale shown below provides a measure of the presence of a desired characteristic.

Characteristic Level (CL)	Degree to Which Characteristic is Present in this Component
3	The desired characteristic is entirely present.
2	The desired characteristic is mostly present.
1	The desired characteristic is somewhat present.
0	The desired characteristic is completely absent.
-1	Characteristics are present which are somewhat detrimental
-2	Characteristics are present which are very detrimental
-3	Characteristics are present which are unacceptable

Evaluation of the Prototype Custodian:

The evaluation of the prototype custodian is provided below. This evaluation is intended to illustrate the technique for evaluating drawer components as well as the drawer itself. A comparison is provided between the maximum possible scores and the scores computed based upon this author's evaluation. The maximum possible Drawer Component Quality (DCQ) for each drawer component was computed by assigning the Characteristic Level (CL) value "3" (the highest possible value) to each characteristic being rated. The maximum Drawer Quality (DQ) was computed using the maximum possible DCQ for each component.

Component Evaluations:

A. Custodian - User Interface Components:

Component Description	Rating Data
1. The custodian provides the only user interface to the toolkit.	

Desired Component Characteristics	EV	CL	EV xCL
1a. No toolkit component can be viewed without custodian assistance.	16	3	48
1b. No toolkit component can be printed without custodian assistance.	16	3	48
1c. No toolkit component can be executed without custodian assistance.	32	3	96
1d. No toolkit component can be moved or copied without custodian assistance.	32	3	96

Highest Possible DCQ:	72
Overall Drawer Component Quality (DCQ):	72
Essentiality Value (EV) for component:	32

Component Description	Rating Data
2. The custodian provides the capability to easily select toolkit components at any design level.	

Desired Component Characteristics	EV	CL	EV xCL
2a. Selection of toolkit components is possible via a hierarchy of menus.	8	1	8
2b. Selection of components is adaptable to suit user familiarity with the toolkit (eg: fully menu-driven for novice and reduced menu presentation for expert).	16	1	16
2c. The custodian can provide access to all toolkit components.	32	3	96
2d. The custodian provides access to toolkit components with a minimum number of user keystrokes.	16	2	32

Highest Possible DCQ:	54
Overall Drawer Component Quality (DCQ):	38
Essentiality Value (EV) for component:	16

Component Description	Rating Data
3. The custodian provides on-line help and aid to the user at each decision point.	

Desired Component Characteristics	EV	CL	EV xCL
3a. User help is available for the custodian-user interface.	32	1	32
3b. User help is available for each toolkit drawer.	8	1	8
3c. User help is available for each drawer component.	16	1	16
3d. User help is available at the user's request at any time prior to a response to a custodian-generated question.	32	0	0
3e. User help for error resolution is available immediately following any error condition arising from the execution of a toolkit component.	2	0	0

Highest Possible DCQ:	54
Overall Drawer Component Quality (DCQ):	11.2
Essentiality Value (EV) for component:	8

Component Description	Rating Data
4. The custodian provides meaningful feedback concerning user entries.	

Desired Component Characteristics	EV	CL	EV xCL
4a. The custodian echoes user inputs to custodian-generated questions.	2	0	0
4b. The custodian requests user verification for "dangerous" requests (eg: deletion of a file).	16	2	32
4c. The custodian identifies input errors to the nearest word containing the error.	8	0	0

Desired Component Characteristics	EV	CL	EV xCL
4d. Immediately following a user input the custodian advises the user that the input is being processed.	8	2	16
4e. Custodian-generated error messages are complete sentences.	16	2	32
4f. Feedback is provided for each user requested action.	32	2	64

Highest Possible DCQ: 41
Overall Drawer Component Quality (DCQ): 24
Essentiality Value (EV) for component: 16

Component Description	Rating Data
5. The custodian provides logical, syntactical, and physical error detection as well as the ability to recover from errors.	

Desired Component Characteristics	EV	CL	EV xCL
5a. The custodian permits the user to edit an input which contained an error and does not require the user to re-enter the entire input.	16	0	0
5b. The custodian screens all user input arguments for correctness prior to any attempt to execute a toolkit component which requires the input arguments.	16	1	16
5c. The custodian does not cease execution if an error occurs in the user input data.	32	2	64
5d. The custodian does not cease execution if a component aborts execution due to an error.	32	2	64
5e. The custodian always ceases execution in a "graceful" manner.	32	2	64

Highest Possible DCQ: 76.8
Overall Drawer Component Quality (DCQ): 41.6
Essentiality Value (EV) for component: 32

Component Description	Rating Data
6. The custodian provides the capability to select an interface display that matches user equipment.	

Desired Component Characteristics	EV	CL	EV xCL
6a. The custodian provides the user with the ability to select a terminal definition from a menu of terminals.	16	0	0
6b. The custodian provides a means for the user to test their terminal selection.	16	0	0
6c. The custodian will permit toolkit operation from a non-video terminal.	32	3	96
6d. The custodian permits the use of user-written custom terminal descriptions.	8	3	24

Highest Possible DCQ: 54
Overall Drawer Component Quality (DCQ): 30
Essentiality Value (EV) for component: 16

Component Description	Rating Data
7. The custodian provides full access to user-selectable options in component tools.	

Desired Component Characteristics	EV	CL	EV xCL
7a. The custodian prompts the user for entry of the minimum number of arguments needed to execute a requested tool.	8	2	16
7b. The user is prompted for optional arguments when a tool execution is requested.	8	2	16
7c. No allowable tool option is filtered from the user input stream by the custodian.	32	3	96

Highest Possible DCQ: 48
Overall Drawer Component Quality (DCQ): 42.7
Essentiality Value (EV) for component: 32

Component Description	Rating Data
8. The custodian provides the ability to identify user-selectable data required by component tools.	

Desired Component Characteristics	EV	CL	EV xCL
8a. The custodian permits the user to identify files containing data for use during tool execution.	32	3	96

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	96
Essentiality Value (EV) for component:	32

Component Description	Rating Data
9. The custodian provides an interface that is simple enough to service very inexperienced users and flexible enough to meet the demands of expert users.	

Desired Component Characteristics	EV	CL	EV xCL
9a. The custodian permits the user to identify their level of toolkit expertise.	32	3	96
9b. The custodian considers the user-identified level of toolkit expertise each time a message is sent to the user terminal.	16	1	16

Highest Possible DCQ:	72
Overall Drawer Component Quality (DCQ):	56
Essentiality Value (EV) for component:	8

Component Description	Rating Data
10. The custodian provides selectable levels of input assistance.	

Desired Component Characteristics	EV	CL	EV xCL
10a. The custodian generates help messaqes based upon the user-identified level of toolkit expertise.	32	0	0

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	0
Essentiality Value (EV) for component:	8

Component Description	Rating Data
11. The custodian provides meaningful feedback during the execution of toolkit components.	

Desired Component Characteristics	EV	CL	EV xCL
11a. The custodian signals the start of execution of any toolkit component.	16	2	32
11b. The custodian signals the completion of the execution of any toolkit component.	16	2	32
11c. The custodian advises whether or not any toolkit component execution was successfully completed.	32	1	32
11d. The custodian signals the start and finish of every custodian activity.	2	0	0

Highest Possible DCQ:	49.5
Overall Drawer Component Quality (DCQ):	24
Essentiality Value (EV) for component:	8

Component Description	Rating Data
12. The custodian provides command names which may be customized for user convenience.	

Desired Component Characteristics	EV	CL	EV xCL
12a. The custodian permits the user to identify a series of ASCII characters to be associated with a user-selected name.	32	0	0
12b. The custodian provides automatic substitution of user-specified characters whenever a user-defined command is encountered.	32	0	0

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	0
Essentiality Value (EV) for component:	2

Component Description	Rating Data
13. The custodian provides the ability to identify a file of commands which are to be executed as though they were real-time user inputs.	

Desired Component Characteristics	EV	CL	EV xCL
13a. The custodian will read a user specified file and examine each line of the file as though it was a toolkit command sequence.	32	0	0
13b. The custodian executes valid commands extracted from a user-specified file as though they were input directly from the user terminal.	32	0	0

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	0
Essentiality Value (EV) for component:	2

Component Description	Rating Data
14. The custodian provides the ability to tailor output messages to include user specified data.	

Desired Component Characteristics	EV	CL	EV xCL
14a. The custodian provides the user with the ability to select additional message data.	32	0	0
14b. The custodian includes user-selected additional output data in all custodian-generated messages to the user.	32	0	0

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	0
Essentiality Value (EV) for component:	2

Component Description	Rating Data
15. The custodian provides verification that the data and tools necessary to fulfill user requests are available to the toolkit.	

Desired Component Characteristics	EV	CL	EV xCL
15a. The custodian verifies the existence and availability of user-owned files needed to execute a toolkit component.	32	0	0
15b. The custodian verifies the existence and availability of toolkit-owned files needed to execute a toolkit component.	32	3	96
15c. The custodian verifies the existence and availability of toolkit components requested by the user.	32	3	96
15d. The custodian verifies the existence and availability of toolkit components needed by other toolkit components during execution of a user-requested operation.	16	0	0

Highest Possible DCQ:	84
Overall Drawer Component Quality (DCQ):	48
Essentiality Value (EV) for component:	32

B. Custodian Access Control Components:

Component Description	Rating Data
1. The custodian controls access to the toolkit, individual drawers, and individual drawer components on a user-to-user basis.	

Desired Component Characteristics	EV	CL	EV xCL
1a. The custodian assigns an access level code to each user of the toolkit.	32	3	96
1b. The custodian-assigned access-level code clearly identifies each user's access to any toolkit component.	32	2	64
1c. The custodian maintains lists of users separated by their toolkit access authorizations.	16	3	48

Desired Component Characteristics	EV	CL	EV xCL
1d. The custodian permits the users with authorization to modify toolkit components with the ability to modify the user access lists.	16	3	48

Highest Possible DCQ:	72
Overall Drawer Component Quality (DCQ):	64
Essentiality Value (EV) for component:	32

Component Description	Rating Data
2. The custodian verifies a potential user's access authorizations in a manner that does not reveal the method used to determine access nor the access authorizations of other toolkit users.	

Desired Component Characteristics	EV	CL	EV xCL
2a. User access lists are accessible only to those users authorized to change the lists.	32	3	96
2b. User access lists are not displayed or referenced in any message during access verification	32	3	96

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	96
Essentiality Value (EV) for component:	32

Component Description	Rating Data
3. The custodian provides an audit trail of repeated attempts to access the toolkit by an unauthorized user.	

Desired Component Characteristics	EV	CL	EV xCL
3a. The custodian can determine if a user access list has been compromised.	32	0	0
3b. The custodian stores information which is sufficient to identify any unauthorized user who attempts to access the toolkit.	32	3	96

3c. The list of unauthorized users is maintained with the same security constraints as the lists of authorized users.	16	2	32

Highest Possible DCQ: 80
Overall Drawer Component Quality (DCQ): 42.7
Essentiality Value (EV) for component: 8

Component Description	Rating Data
4. The custodian maintains a history of all toolkit sessions.	

Desired Component Characteristics	EV	CL	EV xCL
4a. The custodian maintains a log of all user activity, on a user-by-user basis, during each toolkit session.	16	3	48
4b. The toolkit history identifies each execution of the toolkit on a user-by-user basis.	16	3	48
4c. The toolkit history identifies the date and time that each user session began and ended.	32	3	96
4d. The toolkit history identifies each toolkit component executed during a user session.	16	3	96
4e. The toolkit history identifies the date and time that each toolkit component execution began and ended.	16	0	0
4f. The toolkit history identifies the status of the system load and storage capacities at the beginning and end of each user session.	8	3	24
4g. The toolkit history identifies the number of times the toolkit has been accessed during any particular period of time.	16	2	32
4h. The toolkit history identifies the number of times any particular user or group of users has accessed the toolkit during any particular period of time.	16	2	32

Desired Component Characteristics	EV	CL	EV xCL
4i. The toolkit history identifies the number of times that any particular toolkit component has been accessed during any particular period of time.	16	2	32
4j. The toolkit history identifies the date that any particular toolkit component was last modified.	32	0	0

Highest Possible DCQ: 55.2
Overall Drawer Component Quality (DCQ): 40.8
Essentiality Value (EV) for component: 32

Component Description	Rating Data
5. The custodian controls access to each drawer component.	

Desired Component Characteristics	EV	CL	EV xCL
5a. No drawer component may be executed without custodian assistance.	32	3	96
5b. No drawer component may be printed without custodian assistance.	16	3	48
5c. No drawer component may be viewed without custodian assistance.	16	3	48
5d. No drawer component may be moved or copied without custodian assistance.	32	3	96
5e. No drawer component may be changed without custodian assistance.	32	0	0

Highest Possible DCQ: 76.8
Overall Drawer Component Quality (DCQ): 57.6
Essentiality Value (EV) for component: 32

C. Information Control Components:

Component Description	Rating Data
1. The custodian controls the flow of information into, within, and out of the toolkit.	

Desired Component Characteristics	EV	CL	EV xCL
1a. All information required for the execution of any toolkit component either resides in the toolkit, is identified by the user in response to custodian questions, or is a integral part of the operating system of the computer supporting the toolkit.	32	2	64
1b. The existence and availability of all information needed to execute a toolkit component is verified by the custodian prior to the execution of the component.	16	1	16

Highest Possible DCQ:	72
Overall Drawer Component Quality (DCQ):	40
Essentiality Value (EV) for component:	32

Component Description	Rating Data
2. Input data is analyzed to ensure that it is in a format which is compatible with the toolkit formats or in a format which is acceptable to the data translation tools available in the toolkit. Improperly formatted data is rejected.	

Desired Component Characteristics	EV	CL	EV xCL
2a. User-identified data, needed for component execution, is checked to ensure it is in a format compatible with the component or with a toolkit data translation tool.	32	0	0

Desired Component Characteristics	EV	CL	EV xCL
2b. Unacceptable input data is rejected and the reason for rejection is identified to the user.	32	1	32

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	16
Essentiality Value (EV) for component:	32

Component Description	Rating Data
3. All user input data is treated as temporary or transient data.	

Desired Component Characteristics	EV	CL	EV xCL
3a. All non-resident, user-identified data needed for the execution of a custodian action or toolkit component is copied to the toolkit temporary storage drawer prior to utilization.	16	0	0
3b. Data residing in the toolkit temporary storage drawer is removed when no longer needed for the execution of a toolkit component.	16	1	16

Highest Possible DCQ:	48
Overall Drawer Component Quality (DCQ):	8
Essentiality Value (EV) for component:	8

Component Description	Rating Data
4. User requests to add, delete, or modify permanent toolkit data are stored by the custodian for review by the person(s) responsible for the toolkit configuration.	

Desired Component Characteristics	EV	CL	EV xCL
4a. User requests to modify the toolkit data are stored in the toolkit temporary storage drawer.	32	0	0
4b. When the toolkit is accessed by a user who is authorized to modify the toolkit, user-requested toolkit modifications are presented to that user prior to the execution of any toolkit component.	16	0	0

Highest Possible DCQ:	72
Overall Drawer Component Quality (DCQ):	0
Essentiality Value (EV) for component:	32

Component Description	Rating Data
5. Once data has been moved to permanent storage, the custodian prevents modification of the data by any user in subsequent design sessions.	

Desired Component Characteristics	EV	CL	EV xCL
5a. Only those users with proper access levels may view, copy, or modify data residing in the toolkit permanent storage drawer.	32	3	96

Highest Possible DCQ: 96
Overall Drawer Component Quality (DCQ): 96
Essentiality Value (EV) for component: 32

Component Description	Rating Data
6. Once acquired, user input data is internally controlled and translated as needed for tool use.	

Desired Component Characteristics	EV	CL	EV xCL
6a. The custodian is aware of the data formats required by each of the toolkit components.	32	0	0
6b. User access to data stored in the toolkit temporary storage drawer is denied while that data is being used by a toolkit component.	16	2	32

Highest Possible DCQ: 72
Overall Drawer Component Quality (DCQ): 16
Essentiality Value (EV) for component: 16

Component Description	Rating Data
7. Access to any and all temporary or permanent data, either by the user or by a resident tool is controlled by the custodian.	

Desired Component Characteristics	EV	CL	EV xCL
7a. Access to all resident data by a user is controlled by the custodian.	32	3	96
7b. Access to all resident data by a toolkit component is controlled by the custodian.	32	1	32

Highest Possible DCQ: 96
Overall Drawer Component Quality (DCQ): 64
Essentiality Value (EV) for component: 32

Component Description	Rating Data
8. All data output during a design session is delivered via the custodian.	

Desired Component Characteristics	EV	CL	EV xCL
8a. No data is delivered to a storage area outside the toolkit unless the storage area is identified and approved by the custodian. Data paths which are integral to toolkit component structures are approved by the custodian.	32	2	64

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	64
Essentiality Value (EV) for component:	32

D. Drawer Sequence Control Components:

Component Description	Rating Data
1. The custodian controls the sequence in which toolkit drawers are opened and closed. Toolkit components which access other toolkit drawers due to their integral structure are considered to be controlled by the custodian.	

Desired Component Characteristics	EV	CL	EV xCL
1a. No toolkit drawer is accessed unless access is accomplished via the custodian or due to the integral structure of a toolkit component.	32	3	96

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	96
Essentiality Value (EV) for component:	32

E. Sequence Control Components:

1. The custodian controls the sequence in which toolkit components are executed. Toolkit components which execute other toolkit components due to their integral structure are considered to be controlled by the custodian.

Desired Component Characteristics	EV	CL	EV xCL
1a. No toolkit component is executed unless execution is accomplished via the custodian or due to the integral structure of a toolkit component.	32	3	96

Highest Possible DCQ: 96
Overall Drawer Component Quality (DCQ): 96
Essentiality Value (EV) for component: 32

F. Parameter Control Components:

Component Description	Rating Data
1. The custodian controls the parameters used during the execution of component tools.	

Desired Component Characteristics	EV	CL	EV xCL
1a. Whenever possible the custodian assigns default component execution parameters prior to requesting user-specified parameters.	2	0	0
1b. User specified parameters have precedence over custodian-assigned default parameters.	32	0	0
1c. The value of all parameters needed for the execution of a toolkit component are established via custodian action prior to the execution of any component.	32	1	32

Highest Possible DCQ: 66
Overall Drawer Component Quality (DCQ): 10.7
Essentiality Value (EV) for component: 16

G. Toolkit Modification Components:

Component Description	Rating Data
1. The custodian controls additions and modifications to the toolkit drawers and their components.	

Desired Component Characteristics	EV	CL	EV xCL
1a. No addition can be made to the toolkit without the assistance of the custodian.	32	2	64

Desired Component Characteristics	EV	CL	EV xCL
1b. No modification can be made to the toolkit without the assistance of the custodian.	32	1	32

Highest Possible DCQ:	96
Overall Drawer Component Quality (DCQ):	48
Essentiality Value (EV) for component:	32

<u>Overall Drawer Rating</u>

<u>Summary of Drawer Component Qualities (DCQ)</u>

Where: EV = Essentiality Value of the COMPONENT.

n	Component Identifier	Max DCQ	Actual DCQ	Max EV x DCQ	Actual EV x DCQ
1	A1	72	72	2304	2304
2	A2	54	38	864	608
3	A3	54	11.2	432	89.6
4	A4	41	24	656	384
5	A5	76	41.6	2432	1331.2
6	A6	54	30	864	480
7	A7	48	42.7	1536	1366.4
8	A8	96	96	3072	3072
9	A9	72	56	576	448
10	A10	96	0	768	0
11	A11	49.5	24	396	192
12	A12	96	0	192	0
13	A13	96	0	192	0
14	A14	96	0	192	0
15	A15	84	48	2688	1536
16	B1	72	64	2304	2048
17	B2	96	96	3072	3072
18	B3	80	42.7	640	341.6
19	B4	55.2	40.8	1766.4	1305.6
20	B5	76.8	57.6	2457.6	1843.2
21	C1	72	40	2304	1280
22	C2	96	16	3072	512

n	Component Identifier	Max DCQ	Actual DCQ	Max EV x DCQ	Actual EV x DCQ
23	C3	48	8	384	64
24	C4	72	0	2304	0
25	C5	96	96	3072	3072
26	C6	72	16	1152	256
27	C7	96	64	3072	2048
28	C8	96	64	3072	2048
29	D1	96	96	3072	3072
30	E1	96	96	3072	3072
31	F1	66	10.7	1056	171.6
32	G1	96	48	3072	1536

Computation of Drawer Quality (DQ)

1) Maximum Drawer Quality Possible:

n = 32

Sum of all Max DCQ x EV = 56108

Using EQ 2:

$$\text{Max Possible DQ} = \frac{56108}{32} = 1753.38$$

2) Actual Drawer Quality:

n = 32

Sum of all actual DCQ x EV = 37553.2

Using EQ 2:

$$\text{Actual DQ} = \frac{37553.2}{32} = 1173.54$$

3) Ratio of Actual DQ to Maximum Possible DQ:

$$\frac{\text{Actual DQ}}{\text{Max DQ Poss}} = \frac{1173.54}{1753.38} = .67$$

4) Actual DQ Percentage of Maximum DQ Possible: 67 %

VITA

Thomas R. Vermillion was born on August 22, 1948 in Wheeling, West Virginia. He graduated from Central Catholic highschool in Wheeling in 1966 and attended Wheeling College until 1968 when he enlisted in the U.S. Air Force. In 1969 he graduated with honors from the Telecommunications Systems Control Course, USAF Technical Training Center, Keesler AFB, Miss. From 1969 through 1977 he performed duties as a Telecommunications Systems Control Technician (AFSC 307x0) at locations in West Germany, the Phillipines, Illinois, Ohio, and Texas. He was selected for the USAF Airman Education and Commissioning Program in 1977 and was assigned to the Ohio State University to pursue an undergraduate degree in Electrical Engineering. While at Ohio State, he received the Schaffstall scholarship for academic excellence. Ohio State awarded him a Bachelor of Science Degree in Electrical Engineering in June 1980. Upon graduation from Ohio State, he entered the USAF Officer Training School and graduated as a member of the Honor Squadron and the Honor Flight in September 1980. Upon graduation, he was commissioned a second lieutenant in the USAF and assigned to Headquarters, Electronic Security Command, Kelly AFB, Texas where he worked as a Command Control Communications Countermeasures Engineer. In June 1983, he entered the Air Force Institute of Technology to pursue a Master's Degree in Electrical Engineering. He is a registered Engineer-in-Training in Ohio and is a member of Etta Kappa Nu and Tau Beta Pi.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/84D-68	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson Air Force Base, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Block 19	

12. PERSONAL AUTHOR(S)
Thomas R. Vermillion, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	84 December 14	270

16. SUPPLEMENTARY NOTATION
Prepared in Partial Fulfillment of the Requirements for the Degree of Master of Science

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	CAD (COMPUTER AIDED DESIGN), COMPUTER PROGRAM VERIFICATION, COMPUTER PROGRAMS, INTEGRATED CIRCUITS, C Language.
09	02		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: THE SYSTEMATIC INTEGRATION OF VERY LARGE SCALE INTEGRATED CIRCUIT COMPUTER AIDED DESIGN TOOLS INTO A TOOLKIT OPTIMIZED FOR ACADEMIC APPLICATIONS.

Thesis Advisor: Lt Col Harold W. Carter, USAF

Approved for public release: IAW AFR 190-17.
Lynn E. Wolaver 21 Feb 85
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED X SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Lt Col Harold W. Carter, USAF	22b. TELEPHONE NUMBER (Include Area Code) 513-255-6913 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

The characteristics of an integrated, technically complete, academically-oriented VLSI CAD toolkit are presented. A systematic method to integrate CAD tools is developed and supported through the design of a set of CAD tool evaluation metrics. A prototype VLSI CAD toolkit and its custodian written in the UNIX "C" shell are included.

END

FILMED

5-85

DTIC